# Enabling Privacy by Anonymization in the Collection of Similar Data in Multi-Domain IoT

Renato Caminha Juaçaba Neto, Pascal Merindol, Fabrice Theoleyre

# Enabling Privacy by Anonymization in the Collection of Similar Data in Multi-Domain IoT

Renato Caminha Juacaba Neto[a], Pascal Mérindol[a], Fabrice Theoleyre[a,*]

[a]*ICube Laboratory, CNRS / University of Strasbourg, Pole API, Boulevard Sebastien Brant, 67412 Illkirch Cedex, France*

**Abstract**

The efficiency of Internet of Things (IoT) systems heavily relies on possessing trustworthy and up-to-date data. Thus, data collection represents a key feature in IoT applications and services that rely on data-streams. Reusing the same data-stream for distinct queries and services represents a promising way to save energy and reduce operational cost for the deployment. However, acquiring data from different owners through distinct providers incurs privacy issues. Indeed, data crosses the borders of different systems, and the end consumer may get knowledge on the original producer. In-network data aggregation enables the compression of information and improves the privacy while reducing the network load. However, the same data should not be aggregated several times, to provide correct and consistent operations. In this paper, we provide a pub/sub routing scheme for IoT systems that respects privacy constraints. The routers publish offers that describe the data they can export: they may aggregate several offers to respect the privacy constraints as defined by the producers. Each producer has a disposable ID, only used to detect and avoid overlaps in the data-streams. Our performance evaluation highlights the efficiency of our pub-sub routing solution to construct a valid aggregation (without overlap) distributively, while respecting privacy constraints.

*Keywords:* privacy, multi-domain, data collection, IoT, PubSub, Routing

## 1. Introduction

Internet of Things (IoT) applications and devices are more and more common in our daily life. Smart Cities are expected to implement big data architectures, collecting information through a large network infrastructure, comprising Road Side Units, as well as classical cellular networks [1]. Intelligent parking sharing systems are expected to reduce pollution [2]. Indeed, tracking in real-time the available parking spots helps to guide the vehicles and to reduce their fuel consumption. Smart buildings also rely on IoT to make the environment adaptive, reacting to the inhabitants [3].

In these large-scale IoT infrastructures, collecting data takes a central role. Data must be always promptly available and updated for effective decision-making and action [4]. However, sensors generate measurements that may present sensitive information (e.g., location of a vehicle, energy consumption for a private flat). In a general manner, sensors provide real-time information on real world entities [5]. In a systematical study, Hui *et al.* [6] quantify IoT privacy leakages, by exploiting a mobile network traffic dataset. They conclude that privacy leaks cannot be neglected.

End-users usually collect data to make a decision. These *consumers generate queries to collect data of interest.* These queries describe which data should be collected according to its descriptive *metadata*, e.g., location, data type, and ownership. The metadata correspond to specific *producers (sensors) that send their*

*Corresponding author
*Email addresses:* caminha@unistra.fr (Renato Caminha Juacaba Neto), merindol@unistra.fr (Pascal Mérindol), fabrice.theoleyre@cnrs.fr (Fabrice Theoleyre)

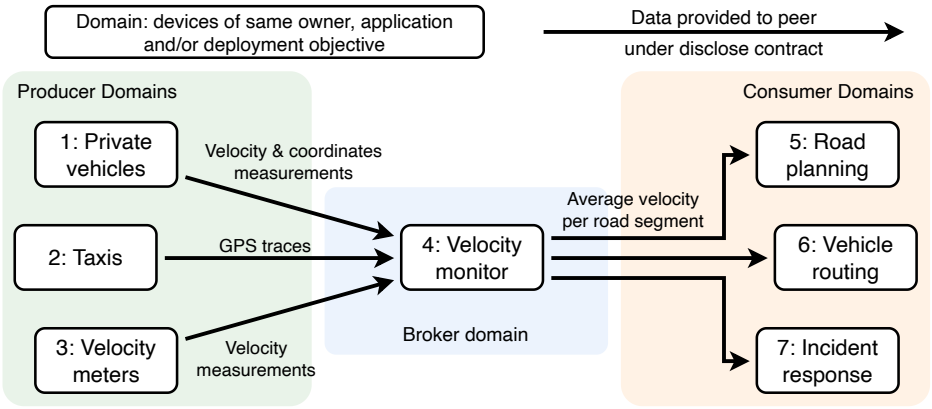



Figure 1: Smart city transport example

*measurements to the consumers.* Consumers mainly query data in the form of streams, where producers will periodically transmit their novel measurements through the network to the consumers. We denote the consumers of a stream as its *subscribers*.

For dense deployments such as smart cities, devices must compete for radio resources. Thus, scalability issues occur [7]. To improve frugality, communication infrastructures as well as sensors may used by multiple applications [8]. For example, taxi companies track the position of their vehicles in real time, while the same data may be reused to infer the speed on the different road segments crossed by their taxis (Fig. 1). Different taxi companies can be seen as different domains and private vehicles can also be seen as individual domains. A trip planning solution may combine this data with a few velocity meters to refine the travel time for its clients [9]. Local authorities may also rely on the same dataset for urban previsions, as well as handling road incidents. Similarly, speed measurements can be acquired by other sources, such as private vehicles and roadside sensors. The multiplicity of sources can improve the decision-making of consumers by diversifying available data.

In the future, we envision infrastructures where distinct systems are owned and managed by different entities and disclose selected information to neighboring systems. In other words, each system encloses a *domain* that discloses selected information to other domains, allowing reuse of data in a controlled manner. Through a multi-domain infrastructure, consumers may query and collect data from producers in other networks and take advantage of data from different owners. By collecting data in such manner, domains enrich the data used by consumers, and both reduce the deployment costs and the amount of redundant devices [10].

However, such a distributed approach implies interoperability and security issues: a domain can now acquire data from multiple concurrent domains, while respecting their respective privacy constraints. In particular, it is necessary to avoid data leaks, and let domains decide their own privacy requirements. These requirements can be defined through the parameters of metrics that quantify levels of anonymization [11]. In this paper, we rely on the k-anonymous metric [12], which dictates that each sample should be indistinguishable from $k-1$ other samples.

We adapt this metric to the context of *aggregation functions* [13]. Mean, min, and max functions are popular examples of such functions. They exhibit the most relevant characteristics of population of samples while drastically decreasing the volume of data. More complex statistical functions such as histograms and density functions also reveal the distribution of values of the samples, while still hiding individual measurements. They spare the network's and device's resources providing directly precomputed data to the consumers [14].

Instead of sharing raw measurements, the system we envision to deploy share aggregated measurements. For instance, the location of a specific vehicle is kept private while the average speed of vehicles at a road segments can be exported. Obviously, to respect privacy, the collection of vehicles must be sufficiently

large to make de-anonymization impractical. Such an aggregation would reach levels of privacy similar to a k-anonymous dataset (if $k$ is the number of samples aggregated per segment).

We argue that Named Data Networking (NDN) [15] seems to be a very attractive way to orchestrate such inter-domain communications. Each domain deploys one or several border routers that offer (publish) data to their peers, i.e., directly connected border routers. Each border router enforces the data processing rules as defined by the consumers. In a previous work [16], we proposed a Named Data Networking (NDN) based tree-based architecture, to publish data and privacy requirements, as well as to handle the subscription phase to construct aggregated data-streams.

In this paper, we aim to enable a multi-domain routing infrastructure where, data is aggregated to respect privacy. We need to carefully aggregate data in this meshed topology: the same data should not be aggregated several times before being forwarded to the consumers. Each router needs to identify overlapping streams (i.e., with producers in common), and to respect the individual privacy requirements of each producer.

The contributions of this paper are as follows:

1. We detail our publication algorithm: each border router selects the data it exports to peering domains, combining different sources of data to respect k-anonymity;

2. We propose a heuristic to merge the data of producers that present the highest similarity in their metadata. Indeed, identifying all possible combinations is intractable, and a border router publishes the most *promising* dataset;

3. We specify a subscription scheme to allow consumers to subscribe to different streams. In particular, it identifies non overlapping data-streams offered by its peers. Besides, it identifies only the metadata of interest;

4. We assess the performance of our aggregation-based approach compared with a centralized approach, which gives an upper bound. We analyze the performance of our algorithms to aggregate measurements, as well as the memory load and processing costs of our technique.

## 2. Related Work

Inter-domain architectures have recently emerged also for the Internet of Things [17]. IoT autonomous systems have to route packets to enable a large-scale interconnection. In this context, privacy is a major concern [18].

### *2.1. Aggregating and forwarding data*

If computation is offloaded to the cloud [19], the infrastructure has to authenticate the sources of the streams, and must provide secure packet forwarding. Moreover, the owner must trust the cloud to store its private data: it would be more relevant to directly exchange data locally, if the consumers and producers are close enough.

Thus, Zhou *et al.* [19] propose to aggregate data that has been generated locally, before forwarding it to the cloud. In-network aggregation saves bandwidth: a single packet may regroup the data from several producers [20]. Propositions exist to maximize the lifetime when constructing such aggregation tree [21]. All the devices are considered equal, and no privacy is considered in the wireless part.

Aggregation relies on a forwarding tree topology: a device aggregates the data from different children before forwarding it.

Existing hierarchies in the network topology can be used to enforce a collection tree [22]: the edge network represents the leaves of the tree, while the fog and the cloud represent upper levels. Clustering algorithms such as LEACH and its successors propose to elect cluster-heads to aggregate data of neighbors [23] in a wireless edge network. While the original version was considering only single hop topologies, extensions have been proposed to handle multi-level clustering [23].

More generically, routing protocols for low power networks create implicitly a tree, rooted most of the time in a *sink*. For instance, each node selects a parent toward a border router (the sink) in RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [24].

On the other hand, PEGASIS [25] builds trees in the form of chains. Neighboring devices elect neighbors in the chain and a leader of the chain. Devices will forward data to the leader through the chain and data is aggregated at every hop. The leader forwards the resulting aggregation to the destination and is periodically replaced in order to equalize energy consumption.

The pubsub scheme that we propose in the following does not rely on a single tree for aggregation. A single tree is not able to satisfy the different aggregation requirements for different queries and criteria. That is, even if a global tree verifying the requirements of all producers exists, more specific queries are then limited to the subtrees of each node and the remaining tree may not be able to satisfy the privacy requirements and match the metadata. Through the use of per producer stream identifiers, we propose to forward the data among systems without data repetition.

### *2.2. Data transformation for Privacy*

In IoT, we need to carefully respect privacy constraints when disclosing data (e.g., measurements of sensors). A sensor may filter some sensitive fields when sharing its measurements. However, removing names and IDs may not be enough to guarantee privacy: an attacker may de-anonymize data if a sufficient number of other fields are present (location, type of sensor, time of activity, etc.) [26].

Data masking consists in removing attributes, in adding noise, or in generalizing (e.g., replacing exact values with intervals) to increase the level of privacy. This technique helps to keep the exact values secret. While these operations are less costly than encryption, they do not provide full opacity [27].

The k-anonymity [12] metric quantifies the privacy level of data. A set of measurements respects the k-anonymity property if none of the measurement can be distinguished via identifiable attributes (e.g., address and age), from at least $k-1$ other values. At worse (when identifiable attributes of an entry are known), the chance of de-anonymization is $\frac{1}{k}$.

Anonymization algorithms for this metric apply generalization and filtering until it is reached. However, data become unusable if they are overused: too generic (transformed) data lose their significance. Utility can be modeled by the loss of information resulting from anonymization. Liang *et al.* [28] minimize this loss, by calculating a normalized ratio between the upper and lower bounds of the generalization. Such minimization problem has been show to be NP-Hard [29]. Approximations such as [30] tackle the problem by partitioning entries into tractable sub-problems.

We are rather interested into data-stream aggregation in IoT networks. Time [31] and count [32] windows are generally applied in such cases [33]. An aggregator node will collect measurements for anonymization until a given number of measurements has been collected or an amount of time has passed. Then, it will anonymize the set of measurements collected and transmit the result. While the former enforces k-anonymity, the latter imposes a maximum delay for anonymization.

Similarly to k-anonymity, a set of field-structured data is $\epsilon$-differential if the removal / addition of any entry does not change the resulting value by more than $\epsilon$ [34]. If a query is very restrictive (very selective metadata), more noise needs to be added since the number of involved values decreases. Differential privacy is particularly relevant for IoT [35] since a collection of *Things* shares data that should not be distinguishable. For instance, measurements for a whole building may be aggregated, so that an attacker cannot distinguish each flat individually [36].

### *2.3. Information Centric Networking & IoT*

Conventional IP networks are focused on creating connections among devices. On the other hand, Information Centric Networks (ICN) focus on the acquisition of data. Here, consumers will query the network for data, and routers will find and forward the required data back to the consumer [37]. This behavior is specially interesting in IoT because it allows devices to request data without the need to discover other devices.

Named Data Networking (NDN) [15] is a promising implementation of ICN due to its features and open source development. Each piece of data has a unique name used to route queries in the network. These names

allow for semantic requests such as */buildingA/temperatures* that denotes the temperature measurements of some building A. Non-semantic names (e.g., data hash) may also be used if the name of the data in question discloses too much information [38].

When some data is required, consumers will issue an *interest* packet with the name of the requested data. Routers will take the name hierarchy created by name components to route packets with longest prefix match tables. In other words, NDN's Forwarding Information Base (FIB) has name prefixes instead of IP address prefixes. Then, NDN introduces the Pending Interest Table (PIT) to allow routers to forward data back to consumers without necessarily knowing the consumer's identity. This table stores the incoming connection of interests that are forwarded to other routers, which essentially stores the reverse path of interests in the network. Then, when routers find the data packet with the requested name, they use the PIT entries to forward data back to consumers.

Consumers and producers have no knowledge of each other while using this query/response exchange. Thus, each node signs its packets, so that any other node can verify data and queries. In particular, a consumer can verify that the data it collects has a correct signature, providing both authentification and integrity. Certificates are also treated as named data that have their name included in signed packets. Since each data chunk is signed, they may be safely cached in the network to save bandwidth: if another consumer is interested in the same data, the cache (aka. Content Store (CS)) will be used to reply immediately. Special attention must be paid however since IoT devices have restricted storage capabilities and are constantly inactive to save battery life. Meaning that special cache strategies must be employed to maximize cache hits in reduced CS and with reduced availability of devices [39].

NDN has also been extended to support aggregation and in-network processing for IoT. Tschudin *et al.* [40] proposed Named Function Networking which added support for lambda expressions which are directly evaluated by routers. Besides, functions may be acquired and cached like named data. The aggregation function to apply is directly specified in the name of the data of interest. In turn, Krol *et al.* [41] propose to implement named functions as lightweight virtual machines. These virtual machines support complex code that routers execute before forwarding data packets. Krol *et al.* propose to execute these virtual machines at routers based on three criteria: (i) how frequently they receive interests packets for them (popularity); (ii) compute delay sensitive computations as close as possible to consumers; (iii) compute bandwidth hungry applications as close as possible to producers. Abidy *et al.* enable in-network processing functions [42] for low power lossy networks where devices disseminate the interest from the sink and iteratively aggregate data from each other. These works provide the in-network feature without imposing privacy constraints, which is what we seek to provide in this work.

### *2.4. Multi-domain IoT*

Interoperability is a key challenge since we face a large collection of independent deployments, each with their own devices and protocols. An IoT network is deployed with one gateway that forwards data from sensors to the Internet. Interconnection may be achieved at the application layer. MQTT or CoAP proxies may cache and exchange data with hosts in the Internet [43]. These proxies can also act as translation gateways [44] which support different protocols to interconnect heterogeneous systems. A proxy may enforce access control rules to protect unauthorized access [45].

Attribute-based access control is the main strategy to define policies [46]. Here, a decision entity decides on whether a consumer has access to a certain data based on a set of attributes that describe the consumer and the data.

Role Based Access Control (RBAC) extends this scheme to grant access based on the roles of the users. Chen *et al.* [47] rely on a trust evaluation algorithm to define who can access to a given data. However, we still need an entity to manage the identities, and a way to detect leaks and misbehaviors to adapt the trust metric. We rather rely on a fixed topology of peering domains, that trust only neighbors.

## 3. Scenario and Problem Statement

In IoT networks, each application commonly deploys its required sensors and actuators. For instance, smart cities rely on induction loops to control traffic lights in order to reduce traffic jams. However, we are

convinced that a sensor may generate data which could be useful for several applications. For instance, the induction loops of a smart city may also be used to count vehicles and to estimate the speed of different road segments. While this flexibility would reduce the deployment cost, we have to deal with multi-owner sensors. In particular, we need to respect privacy constraints and let owners control the data it produces.

We envision an architecture composed of the following elements (illustrated in Figure 2):

a **domain** groups devices based on criteria such as ownership, application, or deployment objective. Within these boundaries, any device can freely exchange data generated locally. In particular, data can be exploited by any application part of this domain. Thus, we do not have any privacy issue when data transits within a domain;

a **border router** forwards data among domains. It enforces the privacy policies when exporting data to peering border routers, in neighboring domains. More precisely, it transmits its available data to peering border routers;

a **dataset** defines a data-stream that can be exported, described by its **metadata**. Metadata are descriptive attributes that characterize dataset (e.g., geolocation, type of measurements). We will later see that this also includes the list of producers that are used to construct the dataset;

an **offer** announces a dataset that can be shared with peering border routers. It is defined by the associated producers, their metadata, and its privacy requirement. It is similar to NDN advertisements, but differs slightly for the subscription phase: a consumer selects a subset of the dataset in an offer. This subset has to respect the privacy requirements of producers in the dataset offered;

a **producer** is a device (aka. sensor) that generates data-streams. A data-stream corresponds to a time-series of measurements. We also denote a domain that exports data of its producers as a *producer domain*;

a **subscriber** is a device that consumes data-streams. More precisely, it constructs a query that specifies its interest, which is transmitted to its border router. Then, our overlay handles the subscription to the data-streams of interest, i.e., that match the query;

a **broker** is a specific border router that collects several data-streams. It can aggregate them and forward this new aggregated data-stream to a neighboring border router, that owns to a different domain.

We model the relationships among domains as a directed graph $G(V, E)$ where each vertex is an IoT domain and an edge $u \to v$ exists if domain $u$ is a provider of data to its peer $v$. An edge represents a trust or commercial relationship among domains. These relationships bind receivers to respect the restrictions of providers (privacy, metadata, etc.). In particular, a receiver can forward a data-stream only if it respects the k-anonymity property as defined by each of the producers. We enforce directional links as some domains may acquire data from another but not share its own data. This is a crucial aspect since domain may further disseminate data (as brokers) without the involvement of the original producer domain. Thus, domain must limit to whom their data is disclosed.

### *3.1. Datasets and Aggregation Functions*

Aggregations are operations that provide us information on the population of a random variable through the use of multiple samples. We model datasets here as multisets denoted $\mathcal{M} = (A, m)$ where $A$ is the set of distinct data values observed and the multiplicity function $m : A \to \mathbb{N}^+$ denotes the number of times the multiset contains each value in $A$. We also define the sum operator to represent the merging of two datasets as $M_i + M_j = (A_i, m_i) + (A_j, m_j) = (A', m') = M'$ such that:

$$A' = A_i \cup A_j \text{ and } \forall x \in A', m'(x) = \begin{cases} m_i(x), & \text{if } x \notin A_j \\ m_j(x), & \text{if } x \notin A_i \\ m_i(x) + m_j(x), & \text{otherwise} \end{cases}$$

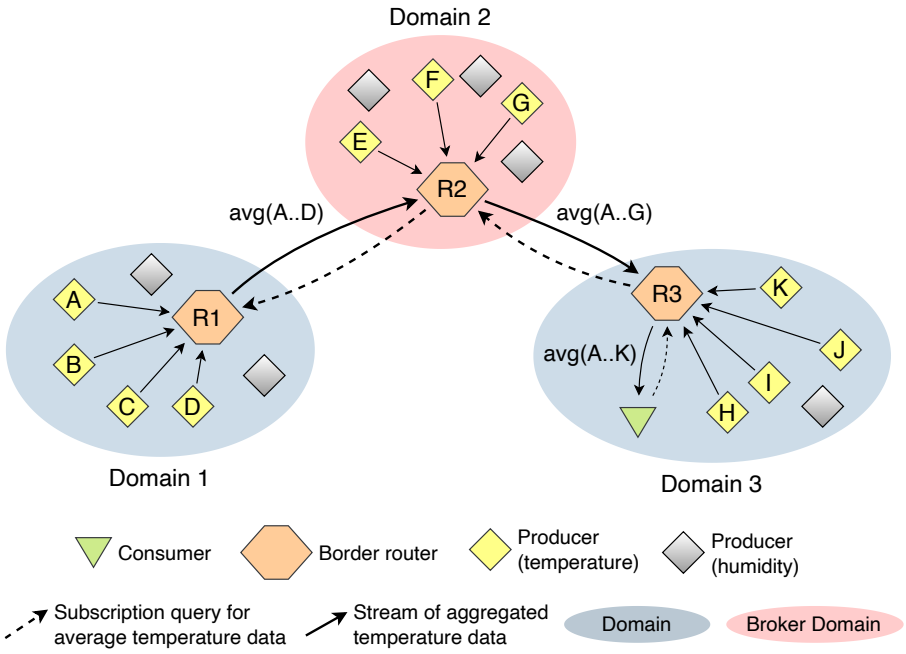



Figure 2: Scenario illustration

Notice that merging two datasets is an anonymous operation since values in $M'$ does not indicate whether it comes from $M_i$ or $M_j$. Thus, it is impossible to identify the original and individual datasets when looking (only) at the resulting merge.

We assume here that all datasets have a set of descriptive attributes. Take for example the dataset of parking sensors deployed in Melbourne to monitor available parking spots[1]. This dataset includes geographical location of more than a thousand sensors that monitor bays of multiple parking lots.

We say that a function $f : \mathcal{M} \to \mathcal{M}$ preserves the sum operation of multisets iif $f(f(\mathcal{M}_i) + f(\mathcal{M}_j)) = f(\mathcal{M}_i + \mathcal{M}_j)$. Commutative and associative functions satisfy such a preservation of the result. Thus, applying such a function to merged or unmerged inputs (and whatever their ordering) does not change the final value.

We say that a function is an aggregation function if it preserves the sum operation and also conserves the cardinality of the resulting dataset. An aggregation function also decreases the level of details to enable k-anonymity: the cardinality remains constant after the aggregation. More specifically, for a given aggregation function $f$ and a set of $n$ input datasets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ such that, for easier notation, $\sum_i^n \mathcal{M}_i = \mathcal{M} = (A, m)$ and $f(\mathcal{M}) = (A', m')$

$$\sum_{x \in A} m(x) = \sum_{y \in A'} m'(y) \; and \; |A| \geq |A'| \tag{1}$$

Average, maximum and minimum are simple statistical functions that match this aggregation with $|A'| = 1$. Such operations result in a multiset with a single value, i.e., $f(\mathcal{M}) = (A', m')$ such that $|A'| = 1$ and $m'(y \in A') = \sum_{x \in A} m(x)$.

Table 1 illustrates different aggregation functions with two datasets $D1$ and $D2$. The multiplicity functions resulting from the minimum ($min$) and average ($avg$) functions indicate the number of input samples used as required in equation 1. Thus, they increase privacy since the individual contributions of each producer is more complex to be inferred. For the average function, a single value is produced, whatever the number of samples.

[1] https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bay-Sensors/vh2v-4nfs

Table 1: Example of three aggregations functions (minimum, average, and histogram functions)

| $D1$ | | $D2$ | | $D1+D2$ | | $min(D1+D2)$ | | $avg(D1+D2)$ | | $hist(D1+D2)$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x \in A$ | $m(x)$ | $x \in A$ | $m(x)$ | $x \in A$ | $m(x)$ | $x \in A$ | $m(x)$ | $x \in A$ | $m(x)$ | $x \in A$ | $m(x)$ |
| 3 | 2 | 7 | 3 | 1 | 2 | 1 | 27 | 5.4 | 27 | 0 | 12 |
| 4 | 1 | 8 | 5 | 7 | 6 | | | | | 5 | 15 |
| 7 | 3 | 3 | 7 | 3 | 9 | | | | | 10 | 0 |
| 8 | 4 | 1 | 2 | 4 | 1 | | | | | | |
| | | | | 8 | 9 | | | | | | |

Data can be aggregated according to the available attributes. A monthly average would, for example, result in $|A'| \leq 12$, i.e., at most one value per observed month. With this method, we can extract controllable approximations such as the standard deviation, median, or quartiles. For instance, we can also use a histogram (*hist*) function, as illustrated in table 1. Here, the value in the resulting set is the lower bound of the bins of the histogram and the value of the multiplicity function is the number of samples in the given bin. Thus, the number of values is upper bounded by the sampling rate of the histogram (in our example, 3 values interspaced by 5 units).

A important property of some aggregation functions is the *sensitivity to repeated values*. Which is the fact that the resulting value changes whether the same data is present multiple times in its input. Formally: $f(\mathcal{M}) \neq f(\mathcal{M} + \mathcal{M})$, for any dataset $\mathcal{M}$.

Some functions such as the min and max functions are not sensitive to repeated values. However, the average and histogram functions are examples of functions which are sensitive. The result of such aggregations is biased to the values which are repeated. For these examples, the more values are repeated, the more bias the result will be.

Any aggregation structure must account for such sensitivity, which is why trees are used to aggregate values distributively. In this work, and in order to remain as general as possible, we prefer to assume that the aggregation functions in use are sensitive to this issue.

### *3.2. Privacy by Aggregation*

Applying an aggregation function to datasets removes details from the resulting dataset as the operation replaces information on individual samples with information on the population of samples ($|A| \geq |A'|$). These operations can be exploited in order to provide levels of privacy similar to that of anonymous datasets. Anonymous data strategies rely on a tradeoff between utility and risk: reinforcing the aggregation decreases the level of information. However, it makes the de-anonymization harder. In other words, there is no upper bound on how much data can be aggregated together [48].

We consider that sufficiently aggregated data is private enough to be disseminated into the network. Peering border routers are not authorized to forward data directly: they must first aggregate data to enforce the privacy requirements of the producers. We denote $req(p)$ as the **minimum aggregation requirement** of each producer $p \in G$. Thus, a border router needs to achieve $req(p)$-anonymity by aggregation. When a dataset is composed of several producers, a border router must respect the highest privacy requirement among all of them. Let $P$ be a set of producers part of a dataset, all border routers must respect the following minimum aggregation requirement:

$$req(P) = Max_{p \in P}\left(req(p)\right) \tag{2}$$

It is worth noting that respecting the minimum aggregation requirements creates constraints on the forwarding scheme. In particular, a producer may specify a very high minimum aggregation requirement, that prevents any peering domain to aggregate it. In conclusion the topology of domains must be sufficiently dense, and/or the minimum aggregation requirement sufficiently low to enable a global dissemination. However, we are convinced that's the cost to pay to respect privacy.

*3.3. Properties*

This paper focuses on creating valid aggregations in a multi-domain environment. The following three properties must be preserved when computing aggregated data:

**P1 Only data of interest must be involved:** Subscribers describe their interest in a query, with a collection of criteria. Our subscription scheme must identify the relevant Matching Producers (MP), i.e., the metadata of these producers match the criteria of the query.

**P2 Enforce non-intersecting datasets:** the aggregation is applied recursively to form a stream. Obviously, the same producer must not appear several times in this aggregation. Else, considering the same sample value multiple times leads to bias.

**P3 Aggregation requirements must be preserved:** the aggregation must respect the minimum privacy requirement of each producer. This way, we enforce k-anonymous datasets [12]. We assume here that we can trust peering domains, i.e., a domain respects the privacy requirements defined by its peers.

## 4. Publication and subscription of aggregated data

Our proposal builds on top of a NDN architecture. While our previous work [16] assumed the domains form physically a tree topology, we propose here a publish/subscribe mechanism able to detect overlaps, and to construct globally consistent streams. Our publish/subscribe scheme proceeds in two phases:

1. **Publication**: routers disseminate the valid combinations of producers. Each combination forms a dataset that respects the minimum privacy requirement of each producer;
2. **Subscription**: the consumer constructs a query that defines its interest, and sends it to a border router of its domain. This border router is in charge of identifying the producers that match the query. It will subscribe to a subset of the matching producers, such that the privacy requirements are respected.

*4.1. Creation of Producer IDs*

Border routers propagate offers during the publication phase, with the dataset, and their metadata. Then, during the subscription, they parse the offers it received in order to identify those that match the interest of the consumer. To avoid using the same data twice in the aggregation, we need to identify the producers. More precisely, we need to identify that a data which is part of several offers is generated by the same producer.

Our method maintains correct aggregation through the use of *producer IDs* that are used by border routers to identify the data used in aggregations. To maintain the privacy, a border router should not be able to identify the actual producer behind a producer ID. It should just use this ID to detect overlaps in the aggregation, and thus, to prevent this kind of situation. These IDs are locally created by each producer via the use of the hash of the network address and the metadata. If we assume the hash function is well-chosen, a border router cannot revert the producer ID to obtain its real address. Producer IDs help us to provide both properties **P1** and **P2**.

We may have collisions in the hash function: the same producer ID corresponds to two different actual producers. The fingerprint has to be long enough to limit the number of collisions. Besides, collisions do not jeopardize the safety of our solution: privacy requirements are still enforced, and the aggregation is still consistent. The aggregation tree would in that case include only one of the producers that collide. Thus, a collision just reduces the number of producers we can include in the aggregation.

Highly descriptive metadata may be attached to a dataset, such that an attacker could identify the producers IDs with the metadata. For instance, if the exact geolocation is included, an attacker may detect a private flat that generated the data. Thus, we assume that some level of anonymization is applied, such that real entities cannot be associated with IDs.

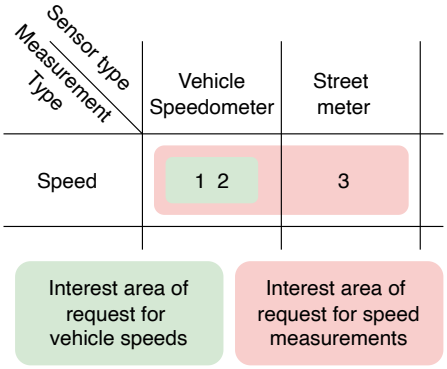



Figure 3: Illustration of vertex location in the descriptive space of our smart city transportation example

### *4.2. Metadata similarities*

When a border router has to construct the list of datasets to publish, it must merge/aggregate some of them, else, the number of offers would increase exponentially. We propose to merge datasets that contain **similar** producers. More precisely, we define the *metadata similarities* as the distance in the descriptive space. Such distances are easy to measure for continuous attributes (e.g., geolocation). For categorical attributes (e.g., device types), we need another quantification function [49]. We assume here a similarity function $\Delta f$ exists, that quantifies the similarity between two datasets.

In Figure 1, domains 1, 2, and 3 produce data with metadata containing their geolocation, as well as the measurement and sensor type, its accuracy, etc. Domains 1 and 2 have very similar metadata: a border router will merge preferentially domains 1 and 2. Domain 3 will likely be merged with more similar domains.

### *4.3. Publication phase*

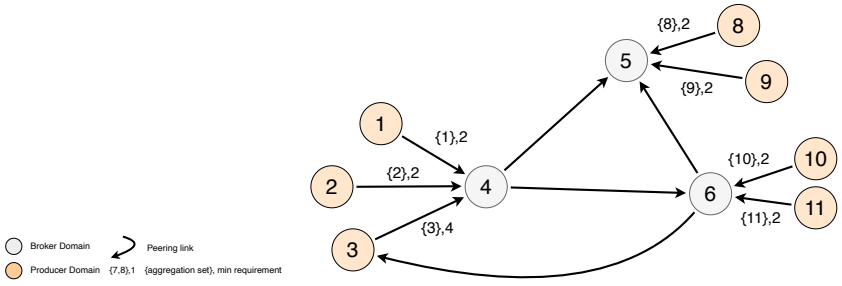
Each border router has to identify the datasets it can publish, i.e., share with its peers. An aggregation offer is a pair $o = (AS, r)$ where **Aggregation Set (AS)** denotes the set of producer IDs that are being offered in offer $o$ and $r$ denotes the minimum requirement of these producers in offer $o$. Such an offer denotes that any combination of producers $C \subseteq AS$ may be enabled as an aggregated stream as long as it is compliant with the minimum requirement, i.e., $|C| \geq r$. On a high level, *publication* consists in filtering and/or transforming **aggregation offers** from incoming neighbors (**input offers**) and exporting aggregation offers to outgoing neighbors (**output offers**).

Let us consider Figure 4, a slightly modified version of the use case described in Figure 1. A producer domain generates data (with the corresponding producer ID), that is exported through its border router. Similarly, a broker domain comprises a border router which is peering with other border routers of peering domains. For the sake of simplicity, we use indistinctly the terms *producer* (resp. *broker*) and *producer domain* (resp. *broker domain*).

In Figure 4, we consider producers which may also subscribe to a stream, i.e., being also consumers. More specifically, the edge $6 \rightarrow 3$ illustrates that domain 3 may both produce data (with producer ID 3) and may also consume data offered by domain 6. For example, a private house may represent a domain (domain 6). It sends its energy consumption to its electricity provider, but may also retrieve the amount of energy produced regionally to adapt its behavior (e.g., starting or not a washing machine depending on the energy availability). In conclusion, cycles may exist in the topology, even without bidirectional links among domains (their size can be longer than two).

The domain 1 exports an offer $(\{1\}, 2)$, meaning that it exports data from the producer ID 1, that has to be merged with at least 1 other producer before being exported (to respect a minimum aggregation requirement of 2).

We assume, for simplicity, that the publication happens in a synchronized manner, i.e., messages are exchanged at the same time and instantaneously. It is worth noting that the publication works also if offers




(a) First round

(b) Second round

(c) Third round

(d) Fourth round

Figure 4: Extended smart city transport example and offers published

are exchanged asynchronously: a border router keeps only the offers received in the last announcement for each of its peers. We consider here a discretized time, counting **rounds**. In figure 4a, we specify the first round for which a given offer starts to be published. At the beginning of each round, a border router receives offers, process them, and publishes them at the beginning of the next round.

*4.3.1. Completing offers to respect privacy requirements*

Producers initially disclose their dataset as an offer with an AS containing only their ID and $r$ being their own (individual) requirement (see figure 4a). Such offer naturally has a requirement larger than the available number of producers ($|AS| = 1 < r$). They correspond to what we name **incomplete offers**. Analogously, a **complete offer respects the minimum requirement ($|AS| \geq r$).**

When a border router receives an input offer which is complete, it can forward it unchanged to its peers, in the output offers. Oppositely, incomplete offers cannot be disseminated: they do not respect the privacy constraints of *some* producers. Border router must extend these incomplete offers before publishing them. However, a border router can merge only non-intersecting input offers. If, after the merge, the aggregated input offers respect the minimum aggregation requirement of each producer, they form a valid output offer.

The minimum aggregation requirement must be equal to the exact cardinality of the union of the Aggregation Set (AS). The requirement of the aggregated offer must be larger than the requirement of each

individual input offer. However, the exact set of producer IDs is selected at the query time. Thus, considering the worst case represents the only one way to enforce the respect of the requirement of each individual input offer.

In conclusion, we guarantee property **P3**. When offers are merged, a peer further in the path cannot identify where data has been aggregated. In particular, it hides the logical topology used to construct the aggregated set. We also hide the privacy requirements of each producers. This anonymity represents one of the strengths of our solution.

Let us consider figure 4: broker 4 receives 3 incomplete offers during the first round. However, in the second round, it is now able to publish a new offer $(\{1, 2\}, 2)$ (Fig. 4b), that is forwarded to peering routers 5 and 6. The routers 5 and 6 cannot disaggregate the stream to retrieve individual measurements of the producers 1 and 2.

*4.3.2. Avoid duplicated aggregated sets*

Classical routing protocols traditionally prevent forwarding loops: if a cycle exists in the routing topology, packets cannot be correctly delivered. Concretely, the same router should not forward twice the same packet. In our architecture, a data loop exists only when the data of the same producer is aggregated several times in the same stream. Thus, each router has to aggregate streams that don't overlap, to guarantee property **P2**.

Besides, a router considers only offers that can complete existing offers in a better way. More precisely, a router considers only the first peer that announces a specific offer. If the same offer is received from another peer later, it is simply discarded since it may trigger a data loop for this offer. The combination of these two mechanisms (i.e., non overlapping aggregated set and to consider only the first received offer) are enough to avoid the creation of any data loop in our routing architecture.

Let us again consider figure 4 to illustrate such specificities. In round 3, domain 5 receives the offers $(\{1, 2\}, 2)$ from domains 4 and 6. Then, at round 4, it may generate the invalid offer $(\{1, 2, 1, 2\}, 4)$ if it would be not invalidated (due to repeated producer IDs). The same will again happen at round 5 (not illustrated here) since domain 6 could generate the invalid offer $(\{1, 2, 3, 10, 11, 10, 11\}, 7)$ In our algorithm, we enforce empty intersections when merging offers.

Authorizing cycles in the physical topology allows us to increase the number of possible offers. Let us focus on the producer 3 which has a large privacy requirement in figure 4:

1. Broker 6 can aggregate the stream from producers 10 and 11, and publishes this offer to its peers (step b);
2. Node 3 receives this offer, that it can forward to 4 (step c);
3. Broker 4 can then aggregate the data from individual producers 1, 2, and 3 with the stream $(\{10, 11\}, 2)$. This novel offer $((\{1, 2, 3, 10, 11\}, 2))$ is sent to its peers 6 and 5 (step d).

It is worth noting that a cycle exists in the physical topology: node 6 first publishes the offer with only 10 and 11, which is augmented by node 4 with producers 1, 2 and 3. However, the data of the same producer is not aggregated several times in the offer: we guarantee property **P2**. A data loop can be easily identified by any broker: two different offers cannot be merged if their aggregated overlapping sets.

*4.3.3. Algorithm to construct complete offers*

Let us define the conflict graph $CG(C^*, E')$ where the vertices $C^*$ are the available combinations of all valid combinations of producers. An edge exists in the conflict graph between two vertices if the corresponding sets intersect $(E' = \{(C, C')|C, C' \in C^* \wedge C \cap C' \neq \emptyset\})$. Figure 5 illustrates the conflict graph for border router 4. Creating novel offers from incomplete offers consist in finding maximal independent sets in the complement of the conflict graph CG: all the offers in an independent set are pairwise disjoint. Thus, creating offers is a NP-Hard problem as an exponential number combinations of offers is possible, and each offer can potentially possess an exponential number of combinations of producers [50].

An exhaustive exploration is too expensive for large instances of the problem. Thus, we rather propose an heuristic, that selects greedily independent vertices in the conflict graph. More precisely, we add randomly

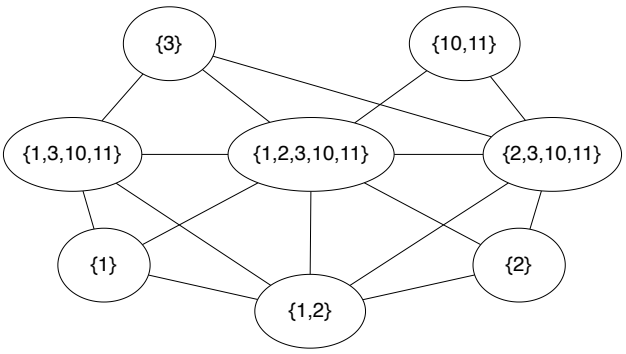



Figure 5: Conflict graph (CG) of the border router 4

vertices in the independent set until all the vertices are either selected or neighbors of a selected vertex. To limit the computation time, we don't consider all the independent sets, and we stop when one of these two conditions holds:

1. when the number of valid output offers exceeds a threshold ($nO_{max}$);
2. when the number of independent sets exceeds a threshold ($nC_{max}$).

This greedy approach prevents easy de-aggregation through the combination of producers. Consider the offers for round 4 in Fig. 4d. In addition to $(\{1, 2, 3, 10, 11\}, 5)$, it is also possible to create $(\{1, 3, 10, 11\}, 4)$ and $(\{2, 3, 10, 11\}, 4)$. If all of these offers are published, a consumer may construct streams that differ only by one producer, such as: $\{1, 2, 3, 10, 11\}$ and $\{1, 3, 10, 11\}$. In that case, the consumer may de-aggregate the two streams to infer the data produced individually by the producer id 2, which is clearly not suitable.

As already discussed in section 4.2, a border router should merge offers with similar metadata. To answer a query, a subscriber selects the producer IDs that match only its interest (Property **P1**). Thus, the offered AS must regroup producers with similar metadata so that we maximize the probability to exploit this offer in the subscription phase. During publication, we limit the creation of offers where the similarity among producers in $AS$ are above a certain minimum threshold value, $\Delta_{min}$, i.e., $\Delta f(AS) \geq \Delta_{min}$.

Algorithm 1 details the actions taken at publication time:

1. We must first identify all producers IDs which are part of incomplete offers. In lines 1 to 4, we find the list of such IDs and their requirement. These incomplete offers are identified to be completed by other existing offers.
2. We initialize in line 5 the counters used for the stopping conditions. This way, we enforce the $nO_{max}$ and $nC_{max}$ limits: when these counters exceed their respective limit, we stop the creation of offers (lines 18 and 19).
3. The main loop of our algorithm (lines 6 to 19) expands offers one by one. We first copy the initial set of IDs into variable $AS^*$ and expand it by adding producers from incomplete offers one by one as the set complies with their minimum requirement. We also verify that for each producer $p$, there is no intersection (i.e., none of the producer IDs $p$ is already contained in $AS^*$), and it is similar enough to the others (line 10). If this condition holds, we merge the two offers (i.e., add it in the independent set, line 11). More precisely, the offer is saved if its producer IDs do not correspond to an existing offer (lines 14 to 16). The new offer must have requirement equal to the size of the set in order to prevent invalid combinations of producers.
4. The algorithm ends by returning an updated set of output offers (line 20). These are the complete offers from $O^+$ in addition to the offer created with the domains local data.

Table 2: Table of notations

| Variable | Meaning |
|---|---|
| $O^-$ | Set of input offers of local node |
| $O^+$ | Output offers of local node |
| $O_n^+$ | Output offers of local node to node $n$ |
| $u$ | Local node |
| $N^+$ | Outgoing neighbors of local node |
| $\Delta f$ | Similarity function |
| $\Delta_{min}$ | Minimum similarity required for offer creation and query matching |
| $nO_{counter}$ | Counter of offer creation |
| $nO_{max}$ | Limit of offers that may be created per incomplete offer |
| $nC_{max}$ | Maximum number of combinations explored during subscription |
| $C^*$ | Set of all valid combinations from offers available to local node |
| $C_{val}(o)$ | Set of valid combinations from a given offer $o$ |
| $AS$ | Aggregated set of given offer |
| $r$ | Minimum requirement of given offer |
| $req(p)$ | Minimum requirement of producer $p$ |
| $\mathcal{I}(nP_{min}, criteria)$ | A query for at least $nP_{min}$ producers that match given metadata *criteria* |
| $MP$ | Set of matching producers to a given query criteria |
| $BS$ | Set of bootstrapper producers (r = 1) |
| $k_{max}$ | Maximum aggregation requirement among producers |

### *4.4. Subscription phase*

When a border router receives a query, it needs to exploit the available aggregation offers received during the publication to answer it. Our subscription algorithm takes as inputs the query and the set of input offers. Then, it needs to explore the combination of offers that maximizes the number of producers it can collect data from. We denote these producers as the answer dataset.

#### *4.4.1. Identifying offers of interest (matching producers)*

A consumer generates a descriptive query, that is transmitted to a border router of the domain. This border router is then in charge of splitting this query into sub-queries, that are forwarded to the peering border routers. In particular, when several sub-queries are created, the border router will aggregate the data-streams when the reply will be received. Recursively, the subscription phase constructs an aggregation in a distributed manner.

A border router can create a list of all known producers, which is the union of all the input offers. Then, it can identify the producer IDs that have the same metadata as its query: they correspond to the Matching Producers (MP). It must construct a valid aggregation that contains the largest subset of MP. The query specifies a minimum number of producers $nP_{min}$ that have to be present in the answer.

We first need to filter the offers of interest. Formally, the border router needs to identify all offers $o = (AS, r)$ that may be used to reply to the query. These offers must contain an aggregated set that includes only matching producers, and that respects the minimum requirement:

$$\exists C \mid C \subseteq AS \wedge C \subseteq MP \wedge |C| \geq r \tag{3}$$

#### *4.4.2. Identification of non-intersecting offers*

The challenge is then to identify offers that don't intersect, and such that their union maximizes the number of producers. Indeed, we cannot have intersecting offers in the combination, else, the data from the same producer may be aggregated several times, creating statistical bias (property **P2**). We have first to compute the set of matching producers (i.e., their metadata matches the query). Then, we try to identify a union of offers that:

**Algorithm 1:** Generating output streams from input streams

**Data:** Input offers store $O^-$, Previous output offers $O^+$, Similarity function $\Delta f$, Minimum similarity for offer creation $\Delta_{min}$, Limit of combination exploration $nO_{max}$, and Limit of offer creation $nO_{max}$.

**Result:** Updated output offers $O^+$.

1 $iAS \leftarrow \emptyset$ // Find producer IDs in incomplete single offers from neighbors and their requirements
2 **foreach** $(AS, r) \in O^- \mid r > 1 \wedge |AS| = 1$ **do**
3 $\quad iAS \leftarrow iAS \cup AS$
4 $\quad p \in AS, r_p = r$ // Store individual requirements
5 $nO_{counter} \leftarrow 0 \wedge nC_{counter} \leftarrow 0$ // Initialize counters of offer creation and combination exploration
6 **foreach** $(AS, r) \in O^+$ **do** // Begin expanding offers
7 $\quad AS^* \leftarrow AS$ // Copy starting set of producers to expand on
8 $\quad$ **foreach** $p \in iAS$ *in increasing order of requirements* **do**
9 $\quad\quad$ **if** $|AS^*| + 1 \geq r_p$ **then** // If set is large enough for next producer ID,
10 $\quad\quad\quad$ **if** $p \notin AS^* \wedge \Delta f(AS^*, \{p\}) \geq \Delta_{min}$ **then** // datasets are disjoints, and similar enough
11 $\quad\quad\quad\quad AS^* \leftarrow AS^* \cup \{p\}$
12 $\quad\quad$ **else**
13 $\quad\quad\quad$ **break** // Stop adding producers if set has not grown enough
14 $\quad$ **if** $AS^* \neq AS$ **then**
15 $\quad\quad O^+ \leftarrow O^+ \cup \{(AS^*, |AS^*|)\}$ // New offer from extended set of producers
16 $\quad\quad nO_{counter} \leftarrow nO_{counter} + 1$
17 $\quad nC_{counter} \leftarrow nC_{counter} + 1$
18 $\quad$ **if** $nO_{counter} > nO_{max} \vee nC_{counter} > nC_{max}$ **then** // Stop iteration if any limit is reached
19 $\quad\quad$ **break**
20 **return** $\{(AS, r) \in O^+ \mid r \leq |AS| \vee$ *is local data*$\}$ // Only announce complete offers

i) contain matching producers;

ii) respect the privacy requirement of each offer;

iii) are disjoint (i.e., do not intersect).

Maximizing the number of producers in this combination allows the consumer to collect a richer dataset. Similarly to the publication problem, we must find maximal cliques in the conflict graph in order to maximize the number of producers offered to the consumer. Thus, this is an NP-Hard problem as several combinations of producers may be possible.

We stop the heuristic if one of these conditions holds:

1. The combination of producers is above a threshold ($nP_{min}$);
2. The number of independent sets we construct in the conflict graph has to remain under a threshold ($nC_{max}$).

*4.4.3. Exploration algorithm for subscription*

Algorithm 2 proceeds in the following way for the subscription:

1. The border router identifies the set $MP$ that match the metadata criteria defined in the query $\mathcal{I}$ (lines 1 and 2). We denote as $metadata(p)$ the set of metadata of the dataset offered by producer $p$. If the number of matching producers is inferior to the minimum required in the query, we must discard it since we are certain that no answer is valid (line 3);

**Algorithm 2:** Enabling streams to answer a consumer request

**Data:** Output offers $O^+$, Query $\mathcal{I}(nP_{min}, criteria)$, Similarity function $\Delta f$, Minimum similarity $\Delta_{min}$, and Exploration limit $nC_{max}$

**Result:** Matching producer IDs and set of corresponding combinations to satisfy the interest $\mathcal{I}$, or two empty sets if the interest cannot be satisfied

```
1  P_avail ← ⋃_{o∈O+} AS(o)
2  MP ← {p ∈ P_avail | metadata(p) matches criteria}      // Producers that match requested metadata
3  if |MP| < nP_min then
4  |  return (∅, ∅)                                      // Number of producers is not significant for I
5  C* ← ⋃_{o∈O+} {C ∈ C_val(o) | C ⊆ MP}                 // Discover set of all matching combinations of producers
6  nC_counter ← 0                                        // Initialize combination exploration limit
7  foreach C' ⊆ C* do                                    // Consider any combination of the elements in C*
8  |  if nC_max < nC_counter then                        // Verify limit of combination exploration
9  |  |  return (∅, ∅)                                  // Stop if limit has been reached
10 |  else
11 |  |  nC_counter ← nC_counter + 1
12 |  if ∀(X, X') ∈ C', X ∩ X' = ∅ then                  // The aggregated sets don't overlap
13 |  |  if ⋃_{X∈C'} X ⊆ MP then                        // The union contains matching producers only
14 |  |  |  if |⋃_{X∈C'} X| ≥ nP_min then              // The number of producer IDs exceeds a threshold value
15 |  |  |  |  return C'                                // C' represents the set of offers to subscribe to
16 return (∅, ∅)                                        // No matching was found
```

2. We identify the set of matching ID combinations ($C^*$, line 5). They correspond to combinations of producers available from offers which are contained in the Matching Producers (MP) (property **P3**). To simplify our pseudocode, we use the $C_{val}$ function to denote all valid combinations from offer $o = (AS, r)$, i.e., $C_{val}(o) = \{C \subset AS \mid |C| \geq r\}$. In our actual implementation, we do not generate all possible combinations of producers. This is simply a mathematical formulation to simplify the reader's understanding.

3. Similarly to algorithm 1, we limit the exploration of combinations via a counter per query (line 6). This counter is incremented for each failed attempt to find a valid answer. The query is rejected if the limit is exceeded (line 8).

4. We search for a subset, which is any combination $C' \subseteq C^*$ and that respects the following conditions (lines 7 to 15):

   (a) the intersection must be null ($\forall (X, X') \in C', X \cap X' = \emptyset$, property **P2**), so that the same producer IDs is not aggregated several times at subscription (i.e., it is part of only one subscribed offer);

   (b) it only combines offers with producers that match the metadata criteria of $\mathcal{I}$ ($\bigcup_{X \in C'} X \subseteq MP$,property **P1**);

   (c) the number of producers is enough to satisfy the query required significance ($|\bigcup_{X \in C'} X| \geq nP_{min}$).

   If such a combination cannot be found, the query is rejected (lines 9 and 16). Since, in our implementation, we only search until some limit $nC_{max}$ is reached. In practice, the number of permutations is enough to easily cover the exploration limit.

### *4.5. Considerations on NDN implementation*

The publish and subscribe algorithms must answer to three types of interest messages:

**Peering interest:** each border router generates these interests to subscribe to the output offers of its peers. When bootstrapping, all border routers generate one interest for each of its direct peers. Then, the offers received in response to this interest will feed Algorithm 1.

**Intra-domain interest:** they are generated by end consumers, i.e., they correspond to queries of the consumers that specify their meta-data of interest. The consumers push their intra-domain interests to any border router of their own domain. These interests are the ones that feed Algorithm 2.

**Inter-domain interest:** a router that receives an intra-domain interest executes Algorithm 2 that returns the set of offers it has to subscribe to. Then, one inter-domain interest is generated for each offer, and forwarded to the corresponding peer. Such interest includes the list of the producers IDs that need to be aggregated from the offer.

#### *4.5.1. Signatures*

Our approach relies on the **signature of interests**, which is particularly important among peers. Indeed, a border router accepts to share its offers only with its trusted peers. Thus, it rejects any interest with an unknown or incorrect signature. The trust between peers is an human based decision: the owner of the domain decides a priori the domains it is connected to. We thus assume that each domain defines it's own public and private key for authentication of interests. We then also assume that the owner inserts a priori, in the trusted list of each of its border routers, all the public keys of the peers it has selected. Alternatively, a credential management adapted to NDN can be applied [51]: NDN's signature and key locator mechanism is sufficient to distribute public keys[2]

#### *4.5.2. Naming*

For the naming convention, a domain prefix must be included in the interests issued among domains, i.e. for inter-domain interests and peering (output offer) interests. A border router is interested by any offer generated by a selected peer. Thus, it has to include the domain of this peer in its output offers interest. The name of such peering interests, or *output offers interest*, may be for instance `/PeeringDomain/outputOffers`.

For inter-domain interests, a border router selects the offers that must be aggregated. More specifically, it must also select the peering domain that sent first the corresponding set to avoid data loops (see section 4.3.2). Thus, the name of the interest is prefixed by the domain ID of this selected peering border router. In that way, we enforce that each inter-domain interest is properly forwarded to avoid such loops. It is worth noting that the domain ID of the border router that actually aggregates data is unknown: inter-domain interests are forwarded hop by hop based on the offers received from the peers. In that way, privacy is preserved.

*Inter-domain interests* may comprise a large set of producer IDs. Ideally, these IDs would be added (sorted) to the name being queried to directly rely on the NDN's default cache engine: in that way, the NDN caching features can be used without any modification. However, NDN's routing tables would use very long names, increasing significantly the lookup times. For instance, if domains use the MD5 hash function, their IDs would be fixed at 16 bytes each. A large number of IDs would be prejudicial since NDN names are assumed to be a few dozen bytes in total [52, 53].

To avoid such an issue, we propose to use the application parameter field of interest packets. This application field will piggyback the sorted list of producer IDs. Thus, the name in the interest can be composed of an hash of this list; for example, a *inter-domain interest* name would typically be like `/PeeringDomain/hash(ProducerIDs)/type`, where `PeeringDomain` is the domain ID of the peer from which the offer was issued, and `type` is the type of data of interest (e.g., temperature, electricity consumption/production). Such a naming convention allows for re-using the caching strategy of NDN: if the same list of producer IDs is mentioned, the cached data may be offered in response. If nothing exists in the

[2] `https://named-data.net/doc/NDN-packet-spec/0.2.1/signature.htm\#id3`

cache, the peering border router will extract the list of producer IDs in the application parameter field and search for a peer that advertised such offer to forward the matching interest.

Similarly, *intra-domain interests* must include the selection criteria (aka metadata of the queries). We assume that these criteria are implemented as logical expressions on data, similarly to SQL and SPARQL queries [54]. These expressions can be encoded as the final name component of intra-domain interests. Again, NDN's cache engine must be extended to consider this type of criteria. In our previous work [16] we have discussed how NDN's cache engine can be extended for hierarchical criteria. Our extensions allow the proposed inter-domain architecture to identify partially processed data in cache and apply missing processing functions in order to answer novel queries.

## 5. Performance Evaluation

To assess the performance of our proposal, we evaluate it on dense topologies composed of numerous producers attached to several broker domains. Producers provide their information to multiple border routers (brokers) that acquire the data and in turn disseminate the resulting aggregated information to other peering border routers. We can support any metadata in the queries. However, we assume here that the similarity among the metadata of two producers is simply given by the geographical distance among them. More precisely, the similarity is normalized by the maximal length in the simulation area ($D_{max}$). Thus, the similarity between two sets of producers $P_1$ and $P_2$ can be described as follows:

$$\Delta f(P1, P2) = \frac{\sum_{\forall p_1 \in P_1 \wedge p_2 \in P_2} (D_{max} - d(p1, p2))}{|\{(p_1, p_2) | p_1 \in P_1 \wedge p_2 \in P_2\}|} \tag{4}$$

Where $d(x, y)$ is the euclidean distance between producers $x$ and $y$ and $D_{max}$. Practically, border routers will tend to aggregate offers that correspond to producers located in the same area.

The minimum aggregation requirement of each producer may have a significant impact on the convergence of the publication. Indeed, a large minimum aggregation value implies an increase in the number of incomplete offers, such that it becomes more and more challenging to complete them. To analyze the impact of such a requirement on the efficiency of our scheme, we simply rely on an uniform distribution to explore its effects: each producer selects uniformly its minimum aggregation requirement in the interval $[2, k_{max}]$ where $k_{max}$ is a tunable parameter (limited to 10 in our simulations). We also model a given subset of permissive producers, i.e., having a minimum requirement of 1, so that the system can bootstrap. We denote $BS$ this set of *bootstrappers*.

For each simulation, we generate 30 random graphs composed of two types of vertices:

1. 30 border routers (brokers) belonging to a strongly connected graph. We generate an Erdös-Rényi graph [55] such that each edge has a probability of 0.15% to exist. We discard the graph if it is not strongly connected, to generate another Erdös-Rényi graph;

2. 170 producer border routers are attached to the graph of brokers (that are also border routers). More precisely, we generate 170 random locations for the producers. Then, we connect each producer to its 3 closer border routers (in space). This way, border routers offer data of similar producers according to the metadata considered.

At subscription time, we generate 50 queries randomly while we set the combination exploration limit $nC_{max}$ of our scheme to 100 (i.e., we limit to 100 the number of combinations of offers to explore, cf. Alg. 2). Each query is generated by a border router randomly selected in the graph. Each query selects the producers located in a random rectangular area of interest. In practice, the query selects for the evaluation the $|MP|$ producers (by default 50) closest to a randomly picked location. This mimics well a geographical query. Generally speaking, all the parameters used in our simulations are listed in table 3, with their default value.

Many solutions exist in the literature for Named Data Networking (NDN) and Named Function Networking (NFN). However, our focus is rather on privacy for inter-domain routing. Our solution disseminates offers in the meshed topology of border routers, that aggregate data to respect privacy constraints. In

Table 3: Evaluation parameters

| Simulation parameter | Value unless specified |
|---|---|
| Number of graphs generated | 30 |
| border router vertices | 30 |
| Producer vertices | 170 |
| Edge probability between brokers | 15% |
| Producers metadata model | Location in 2D square with side equal to 1 |
| Brokers connected to each producer | 3 closest brokers in 2D square |
| Queries issued in each graph | 50 to random brokers (uniformly picked) |
| Minimum similarity $\Delta_{min}$ | 0.7 |
| Offer creation limit $nO_{max}$ | 10 |
| Combination exploration limit $nC_{max}$ | 100 |
| Size of matching producer set $\|MP\|$ | 50 |
| Number of bootstrappers $\|BS\|$ | 20 |
| Maximum aggregation requirement $k_{max}$ | 10 |

**Algorithm 3:** Unlimited random walk. Aggregation upper bound achievable for a given request in a given topology.

**Data:** Set of brokers $B$, Set of matching producers $MP$, Requirement $r_p$ of each producer $p \in MP$.
**Result:** Set of producers that are collectable $UW$.

$UW \leftarrow \emptyset$ // Begin considering empty aggregation
$\forall b \in B, P_b \leftarrow MP \cap N^-(b)$ // Find producers which are one hop away from each broker
**while** $\exists b \in B \wedge \exists p \in (P_b \setminus UW) \mid r_p \leq |UW| + 1$ **do** // While there are brokers that can still aggregate
  **while** $\exists p \in (P_b \setminus UW) \mid r_p \leq |UW| + 1$ **do** // Aggregate data from new producers
    $UW \leftarrow UW \cup \{p \in (P_b \setminus UW) \mid r_p \leq |UW| + 1\}$
**return** $UW$

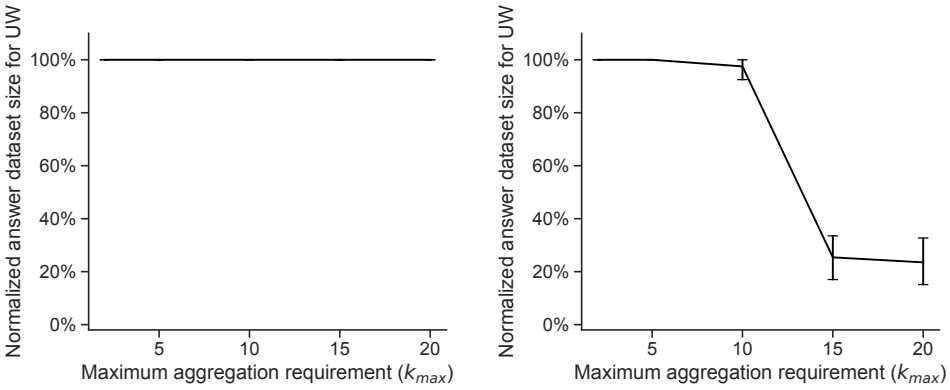
particular, the border routers are able to detect overlaps in the datasets, and to construct distributively a valid aggregation. Thus, we decided to compare the two following approaches:

1. our pub-sub inter-domain routing, described in section 4;
2. a theoretical upper-bound, that could be achieved through a centralized algorithm, which has a complete knowledge of the topology. While this represents an unrealistic assumption that makes its real deployment unrealistic, it serves as a baseline comparison to assess the performance of our solution to construct large and valid aggregations.

### *5.1. Upper Bound for Aggregation: Unlimited Random Walk*

We describe here an upper bound of the aggregation set we can obtain. Obviously, such an ideal approach is inapplicable in realistic situations since it requires a complete central knowledge of the topology, and potentially a complete exploration of each offer combination. Intuitively, an aggregation that respects the minimum requirements of each associated producer can be forwarded to any border router to be completed: the aggregation grows iteratively. Through an *unlimited walk*, we can construct iteratively an aggregation tree as long as we identify at least one remaining border router that can complete it with some locally connected producers. It is worth noting that we have to run this algorithm for each topology, query and distribution of minimum requirements.

Our upper bound (Algorithm 3) maintains the list of producers that have already been aggregated. Initially, the aggregation starts with a single border router having a set of locally connected matching producers




(a) $|MP| = 25$ (b) $|MP| = 10$

Figure 6: Limits of aggregation for restrictive queries

that can be aggregated together while respecting the minimum aggregation requirement. Recursively, we identify a border router that can complete this aggregation with a set of local producers such that:

1. each minimum aggregation requirement is respected;
2. all these local producers are part of the matching producers (i.e., they match the query's metadata).

*5.2. Metrics*

We compute the following metrics for each combination of parameters:

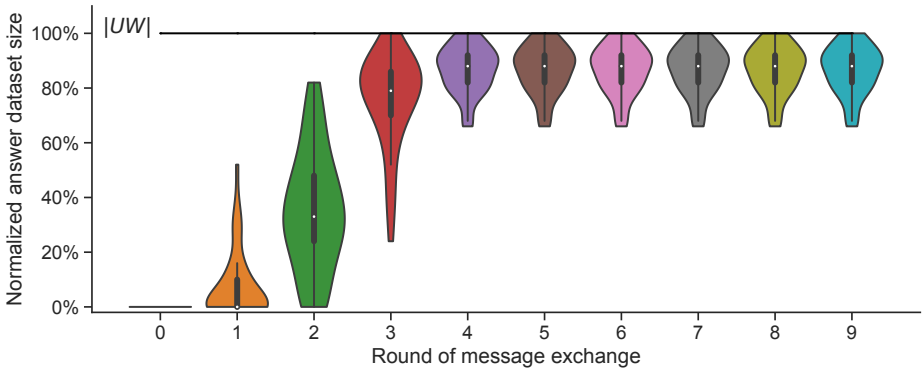
**Normalized answer dataset size:** the number of matching producers that can be aggregated in response to a specific query by using available aggregation offers. Larger is better: the subscriber can acquire larger datasets than expected as long as the set of producers matches the query (it may be included in a larger compliant set). This value is normalized by the number of producers matching the query issued ($|MP|$). We plot distributions in the form of violin plots with our publish-subscribe algorithm. The unlimited walk algorithm does not exhibit a large variability as observed with our heuristic.

**Offer table size:** the number of IDs in the offers available at each border router. Assuming that each ID is stored as a hash, we calculate how many bytes would be required to store all available offers. We compute this sum using hash sizes for the md5 functions.

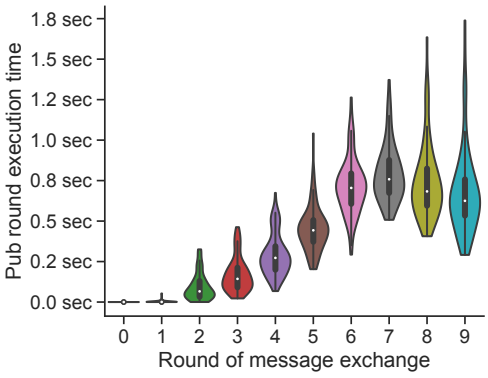
**Processing time:** amount of time taken by our algorithm. For the publication stage, we measure how long each message exchanging round takes. For the subscription stage, we measure how long it takes to find the set of offers to use to answer the query.

**Size of enabled aggregation sets:** number of producers whose data have been aggregated in a stream. We measure, at subscription time, the number of producers in the (enabled) aggregation sets to answer answer some query. This metric tries to quantify anonymity: individual measurements are more and more complex to de-anonymize when they regroup many values.

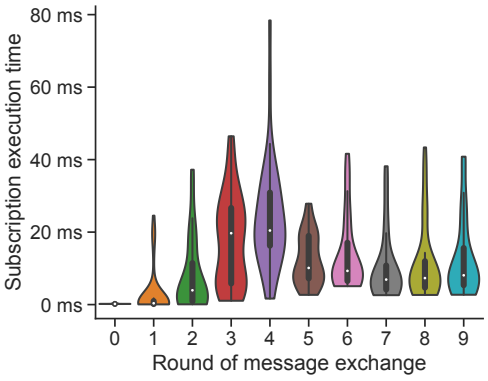
Our figures display these metrics through the use of violin plots. These plots show the distribution of values in the vertical axis. Additionally, our plots include a mini box plot in the middle of each violin to indicate the mean, quantiles and whisker intervals. These graphs should allow the reader to get insight of our population of samples.




(a) Aggregation capacity at different rounds




(b) Duration of each publication round




(c) Time to find streams to answer queries at different rounds

Figure 7: Convergence of the publication step

### *5.3. Boundaries imposed by aggregation requirements*

We first use the random walk algorithm (Alg. 3) to verify the limits of achievable aggregation. Indeed, respecting the minimum requirements limits the possibilities for the aggregation. A producer cannot export data if not enough data can be collected to construct a compliant aggregation ($|AS| > req(p)$).

We measure the normalized answer dataset size (Figure 6). With a large set of matching producers, a border router can combine data from a large set of producers. The probability is higher than enough matching producers are attached to a given border router. Thus, they can aggregate all the matching producers to construct a valid offer. When the query matches 25 producers (Fig. 6a), the subscriber can recursively construct a valid aggregation tree with all the matching producers (normalized answer set = 100%).

On the contrary, using a small minimum aggregation requirements creates more constraints. Indeed, the average number of matching producers attached to the same border router decreases. Thus, the probability is higher to not be able to respect the corresponding aggregation requirements: some of the producers must be discarded. With 10 matching producers (Fig. 6b), the subscribed can still collect the data from all the matching producers when the privacy constraint remains low (i.e., each producer selects randomly its privacy requirement between 1 and 10). However, for highest privacy levels, the aggregation tree fails to collect all the matching producers. Indeed, the privacy requirement would be in the same order of magnitude as the number of matching producers.

### *5.4. Convergence of the publication step*

The centralized random walk can respect any privacy requirement, and it reaches a 100% normalized answer dataset size ( Fig. 7a). Its convergence is immediate since it assumes a centralized, complete knowl-

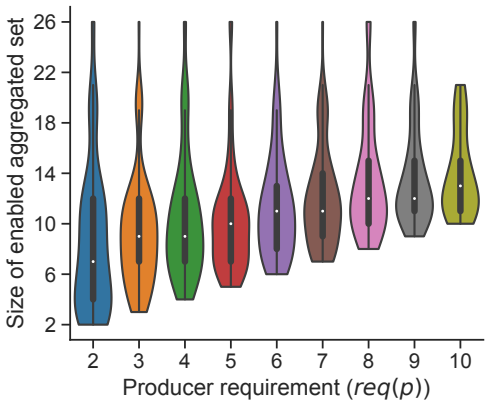



Figure 8: Size of enabled aggregation sets

edge. Even if a producer has a very high privacy requirement, an attached border router can collect an offer that has been generated very far in the network, to aggregate it with the local producer. However, this border router relies on a complete knowledge of the topology, which creates privacy and scalability issues.

For our pubsub scheme, we study the convergence of the publication phase, by measuring the average size of the answer aggregation set at the end of each round (Fig. 7a). Observe that the answer is on average empty with less than 1 round: a router cannot aggregate the data of local and distant producers because of the privacy constraints. In that case, the answer may be empty. In practice, we quantify the execution and subscription times only for non empty answers, in particular Fig. 7b and 7c represent an irrelevant value (zero) with less than 1 round. However, our system converges quickly: the normalized size of the answered dataset grows fast and quickly reaches its maximum. We observe that after only 4 rounds, the border router of the consumer is able to construct an answer aggregation set that respects the privacy constraints with almost 90% of the matching producers. We do not reach a 100% answered dataset size in all the cases since a few producers have too strict privacy requirements: our pub-sub scheme cannot satisfied them.

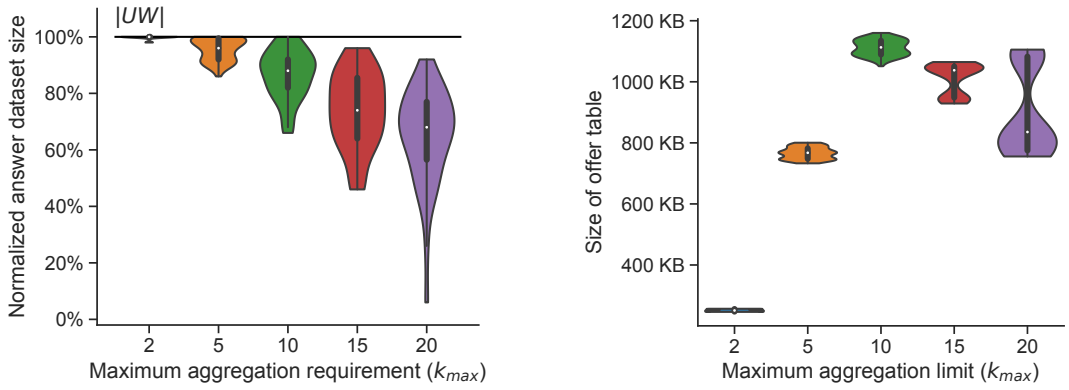
The number of rounds depends both on the eccentricity of the graph and the minimum aggregation requirement, that may generate a larger set of incomplete offers. Larger eccentricities would require more rounds: offers have to be aggregated several times before stabilizing.

Then, we measure the computation time per round for the publication (Fig. 7b). The computation time increases with the number of rounds: each border router has more offers to filter and to complete. However, the computation time even decreases for more than 7 rounds: the incomplete offers are easier to complete thanks to the offers' diversity. In any condition, a border router spends less than 1 second to compute the combination of offers. Since such dissemination is done once and resulting offers are used for any number of queries, this computation cost seems reasonable even for resource constrained devices.

Finally, we also measure the computation time for the subscription (Fig. 7c). While it increases slightly at the beginning since the subscriber has more offers to filter, it converges fast for the same reason as for the publication. After 4 rounds, the computation time tends to decrease since the number of offers increases. Indeed, more offers are available, and a border router can efficiently find offers to aggregate the data of all the matching producers. Thus, our subscription algorithm is efficient to construct a privacy-aware set of producers matching the query. A few dozens of milliseconds are at most required to identify which offers to combine to construct a valid dataset.

### *5.5. Anonymization*

We measure the level of aggregation that our proposed pub-sub scheme reaches (Fig. 8). We measure how many producers are actually aggregated compared to the requirements of each of the producers whose data is aggregated in the enabled stream.




(a) Accuracy of the reply

(b) Size of the offers table

Figure 9: Impact of $k_{max}$

Obviously, the minimum requirement is respected: the size of enabled aggregated sets is at least equal to the requirements of each producer (the size of the aggregated set is always larger than the producer's privacy requirement). However, it is worth noting that on average, much larger aggregated sets are constructed: with a privacy requirement of 2, the aggregated sets comprise the data from more than 6 different producers. In that way, we reinforce the privacy, making the de-anoymization particularly complex.

Indeed, the aggregated set is constrained by the strictest producer. In other words, the aggregated set must be at least equal to the maximum privacy requirement among all its producers. Moreover, our publication scheme iteratively creates offers with increasingly larger aggregation sets. This naturally results in even stricter privacy guarantees.

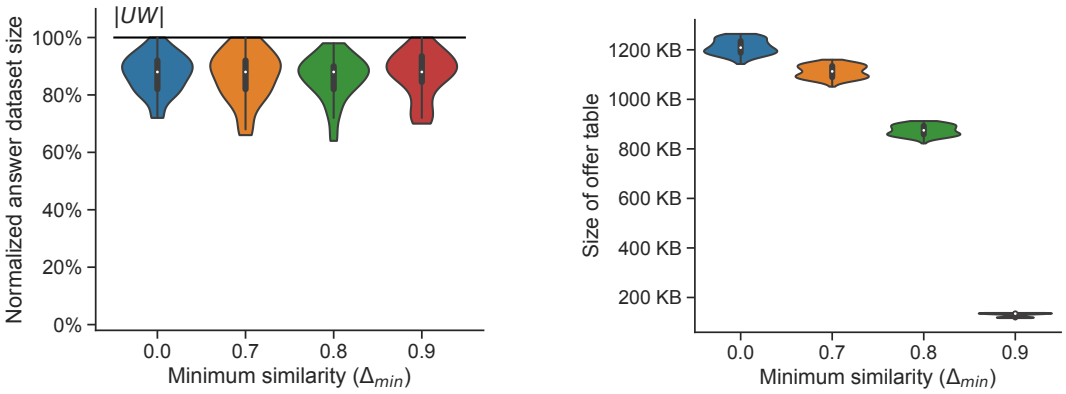
### *5.6. Impact of the privacy requirements*

We now measure the impact of the privacy requirement defined by the producers (Fig. 9). We only report the metrics at the end of the last round, after the system has converged. We first focus on the dataset size (Fig. 9a), we see the behavior that we initially saw with our upper bound resources (sec 5.3). Imposing larger requirements, statistically reduces the achievable answer aggregation set. Indeed, we create probabilistically more privacy constraints, and we restrict the number of producers from which the subscriber can collect data. However, even with very strict constraints, the consumer can subscribe to a significant set of producers, that have probabilistically a smaller minimum aggregation requirement. Thus, our system is robust to heterogeneous privacy constraints.

Let us focus now on the size of the offers' table for each border router (Fig. 9b). The number of offers is very small when privacy is straightforward ($k_{max} = 2$): any border router is able to combine one or two local producers. Thus, no incomplete offer exists, border routers do not need to explore and construct complex sets of offers. A few offers for each couple of attached producers is sufficient to cover the whole dataset. More privacy means a larger number of offers to mutually combine at first. Inversely, a very high privacy implies that incomplete offers at some point cannot be combined, and are not announced. The number of offers present in the table decreases.

### *5.7. Impact of similarity*

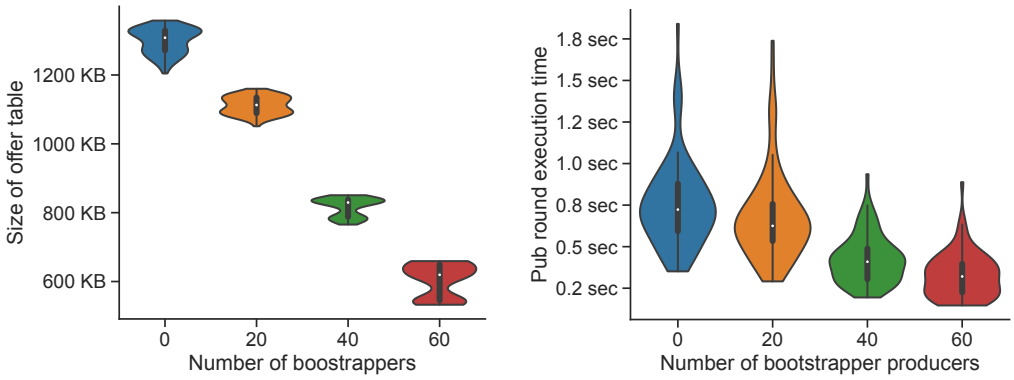
Let us now focus in Figure 10 on the impact of the similarity metric, and in particular the similarity threshold value in Alg. 2. As show in figure 10b when the border router does not consider similarity when merging different offers ($\Delta_{min} = 0$), the number of offers in the table is maximum. Indeed, Algorithm 2 merges two offers if their similarity is larger enough ($> \Delta_{min}$). Thus, a border router generates in that case all the possible merges. The number of offers may significantly decrease (by 85%) when considering a very large similarity threshold value when merging offers. In that case, only offers similar enough can be grouped




(a) Accuracy of the reply

(b) Size of the offers table

Figure 10: Effects of $\Delta_{min}$ on metrics




(a) Size of the offers table

(b) Computation time per round for the publication

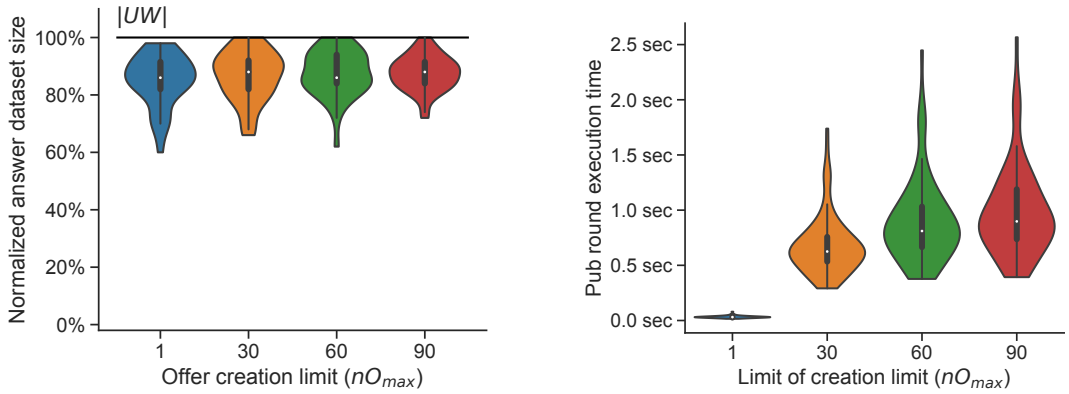
Figure 11: Effects of bootstrapper producers on metrics

together. Else, the border router considers that the offers are not *similar enough* to be part of the same query during the subscription step.

Surprisingly, merging together similar offers has negligible impact on the subscription phase (Fig. 10a). The subscriber is able to construct a dataset of at least the same size, whatever the $\Delta_{min}$ value is. Our method merging the offers according to their similarities is efficient to reduce the number of offers without significant concession on the accuracy. It is worth noting that multi-dimensional similarities may be more challenging as it may limit this effect in more complex and large scale scenarios.

### *5.8. Impact of bootstrapping producers*

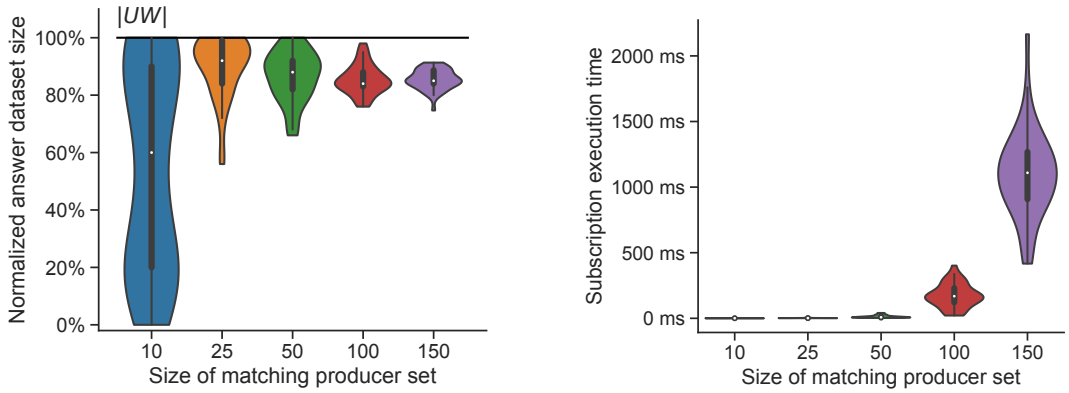
Let us now focus on the impact of presence of bootstrapping producers, i.e., producers without any privacy requirement ($req(p) = 1$). They can be used by any border router to complete their incomplete offers, to allow later the subscriber to select a larger dataset. The bootstrappers reduce the number of offers (Fig. 11a). Indeed, a border router does not need to generate a very large set of offers: combining one incomplete offer with a bootstrapping producer is sufficient. Mechanically, it reduces also the computation time (Fig. 11b), by almost 50% with 60 bootstrapping producers (one third of the producers). It is worth noting that this complexity reduction doesn't impact the accuracy. Indeed, the normalized answer dataset size remains unchanged (and thus not plotted): the subscriber can still use the same dataset for the answer, which is less restrictive than those without bootstrapping producers.




(a) Accuracy of the reply

(b) Computation time per round for the publication

Figure 12: Effects of $nO_{max}$ on metrics




(a) Accuracy of the reply

(b) Computation time for the subscription

Figure 13: Impact of the number of producers that match the query ($|MP|$)

### *5.9. Offers limit creation*

We now quantify the impact of the parameters that limit the exploration during the publication (Alg. 1). When a border router has constructed $nO_{max}$ offers, the exploration stops (line 18). While stopping after having constructed only one offer limits the possibilities, 30 offers are sufficient to provide a good accuracy. Constructing one complete offer from each incomplete offer is sufficient for most cases: since we merge similar offers, the probability to have a relevant offer during the subscription phase is very high.

The computation time (Fig. 12b) is obviously minimal for $nO_{max} = 1$ (creating a single unique complete offer from each incomplete offer). Constructing several complete offers increases the computation time: increasing the accuracy has a cost. However, we quickly reach a maximum: the number of different offers that we can construct is soon reached.

### *5.10. Selectivity of the query*

Finally, we measure the impact of the selectivity of the query on the size of the answer (Fig. 13a). With only few matching producers, the graph is very sparse. Thus, completing these incomplete offers is very challenging, and most privacy requirements cannot be respected. When considering a query, a border router cannot identify a minimum set of producers in an offer that can respect the privacy requirement. The aggregation set contains at most 60% of the matching producers with only 10 matching producers ($|MP| = 10$). The random walk has no topological constraint since it has the complete knowledge of the

topology, and can reach 100% in any case. However, as soon as the number of matching producers becomes reasonable,the graph is denser, and we are closer to the upper bound. When the number of matching producers is very large, the computation time increases significantly (Fig. 13b). Thus, our algorithm has to consider only a subset of the offers, and cannot consider all the possibilities. Indeed, detecting overlaps becomes very expensive. Thus, our solution is efficient for sufficiently selective queries. Else, we should implement more agressive pruning conditions, that would result in a smaller accuracy.

## 6. Conclusion and Perspectives

In this paper we present a publish-subscribe scheme to share aggregated information among anonymous groups of IoT devices (domains). Our method protects the privacy by aggregating data generated by a minimum number of different producers. Only aggregated information is shared, reducing the probability of leaks. We propose an algorithm to construct aggregation offers, that can be disseminated (aka. published) in the network. Each offer specifies the minimum level of privacy that has to be respected. Inversely, we also propose a subscription algorithm, where a consumer sends a query to a border router of its domain. Each border router is in charge of exploiting the offers received from its peers to answer these queries. Recursively, the border routers construct an aggregation tree of data-streams, rooted in the consumer. Since the border routers form a mesh topology, they must safely select the offers to not exploit the data of the same producer(s), aggregated several times. Such a redundant aggregation would lead to statistical bias, and may turn into privacy leakage since the minimum privacy level would not be respected. Our performance evaluation highlights the efficiency of our algorithms to correctly identify the offers to publish and to combine. We show that our method quickly reaches and surpasses required levels of aggregation asked by the producers. We also show that our method can be executed by resource constrained IoT gateways in order to disseminate and collect data in a multi-domain infrastructure.

In a future work, we expect to provide mechanisms to verify that aggregations cannot be de-anonymized throughout the network. Indeed, a border router should not be able to generate overlapping queries, to extract a posteriori the individual measurements of a specific domain (through, e.g., a Gaussian elimination). Filtering may be applied both during publication (not disseminating partially overlapping offers), and during subscription (active streams must concern non overlapping producers and must re-use the same cached data when possible). We also expect to explore how more sophisticated aggregation functions may scale. In particular, we need to investigate how unaggregated data can be cached *en-route*. Even if different aggregation functions are used, a border router could re-use the cached data to generate new data-streams. By judiciously selecting the aggregating border routers, we could reduce both the overhead and the cache size.

## Acknowledgment

This work was supported by the French National Research Agency (ANR) project Nano-Net under contract ANR-18-CE25-0003.

## References

[1] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, S. Guizani, Internet-of-things-based smart cities: Recent advances and challenges, IEEE Communications Magazine 55 (9) (2017) 16–24.

[2] A. M. Said, A. E. Kamal, H. Afifi, An intelligent parking sharing system for green and smart cities based iot, Computer Communications 172 (2021) 10–18.

[3] A. Kumar, S. Sharma, N. Goyal, A. Singh, X. Cheng, P. Singh, Secure and energy-efficient smart building architecture with emerging technology iot, Computer Communications 176 (2021) 207–217.

[4] K. Yasumoto, H. Yamaguchi, H. Shigeno, Survey of Real-time Processing Technologies of IoT Data Streams, Journal of Information Processing 24 (2) (2016) 195–202.

[5] D. E. Kouicem, A. Bouabdallah, H. Lakhlef, Internet of things security: A top-down survey, Computer Networks 141 (2018) 199–221.

[6] S. Hui, Z. Wang, X. Hou, X. Wang, H. Wang, Y. Li, D. Jin, Systematically quantifying iot privacy leakage in mobile networks, IEEE Internet of Things Journal 8 (9) (2021) 7115–7125.

[7] I. Yaqoob, I. A. T. Hashem, Y. Mehmood, A. Gani, S. Mokhtar, S. Guizani, Enabling Communication Technologies for Smart Cities, IEEE Communications Magazine 55 (1) (2017) 112–120.
[8] J. An, F. Le Gall, J. Kim, J. Yun, J. Hwang, M. Bauer, M. Zhao, J. Song, Toward Global IoT-Enabled Smart Cities Interworking Using Adaptive Semantic Adapter, IEEE Internet of Things Journal 6 (3) (2019) 5753–5765.
[9] A. Falek, A. Gallais, C. Pelsser, S. Julien, F. Theoleyre, To re-route, or not to re-route: Impact of real-time re-routing in urban road networks, Journal of Intelligent Transportation Systems 26 (2) (2021) 198–212.
[10] M. Drosou, H. V. Jagadish, E. Pitoura, J. Stoyanovich, Diversity in Big Data: A Review, Big Data 5 (2) (2017) 73–84.
[11] I. Wagner, D. Eckhoff, Technical Privacy Metrics: A Systematic Survey, ACM Computing Surveys 51 (3) (2018) 57:1—-57:38.
[12] L. Sweeney, k-Anonymity: a Model for Protecting Privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (05) (2002) 557–570.
[13] R. K. Yadav, M. Gupta, Data Aggregation Algorithms in IoT: An Organized Evaluation of The Literature, in: International Conference on Smart Systems and Inventive Technology (ICSSIT), IEEE, 2020, pp. 300–304.
[14] S. Abbasian Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, M. Abbasian Dehkordi, A survey on data aggregation techniques in IoT sensor networks, Wireless Networks 26 (2) (2020) 1243–1263.
[15] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, Named data networking, ACM SIGCOMM Computer Communication Review 44 (3) (2014) 66–73.
[16] R. J. Neto, P. Merindol, F. Theoleyre, Transformation based routing overlay for privacy and reusability in multi-domain iot, in: International Symposium on Network Computing and Applications (NCA), IEEE, 2020, pp. 1–8.
[17] S. Park, N. Crespi, H. Park, S.-H. Kim, Iot routing architecture with autonomous systems of things, in: IEEE World Forum on Internet of Things (WF-IoT), 2014, pp. 442–445.
[18] C. Li, B. Palanisamy, Privacy in internet of things: From principles to technologies, IEEE Internet of Things Journal 6 (1) (2019) 488–505.
[19] J. Zhou, Z. Cao, X. Dong, A. V. Vasilakos, Security and privacy for cloud-based iot: Challenges, IEEE Communications Magazine 55 (1) (2017) 26–33.
[20] N. KhadirKumar, A. Bharathi, Real time energy efficient data aggregation and scheduling scheme for wsn using atl, Computer Communications 151 (2020) 202–207.
[21] K. Kalpakis, S. Tang, A combinatorial algorithm for the maximum lifetime data gathering with aggregation problem in sensor networks, Computer Communications 32 (15) (2009) 1655–1665.
[22] S. Sanyal, P. Zhang, Improving quality of data: Iot data aggregation using device to device communications, IEEE Access 6 (2018) 67830–67840.
[23] S. K. Singh, P. Kumar, J. P. Singh, A Survey on Successors of LEACH Protocol, IEEE Access 5 (2017) 4298–4328.
[24] T. Winter, et al., Rpl: Ipv6 routing protocol for low-power and lossy networks, RFC 6550, IETF (2012).
[25] S. Lindsey, C. Raghavendra, Pegasis: Power-efficient gathering in sensor information systems, in: Proceedings, IEEE Aerospace Conference, Vol. 3, 2002, pp. 3–3.
[26] L. Sweeney, Weaving Technology and Policy Together to Maintain Confidentiality, Journal of Law, Medicine & Ethics 25 (2-3) (1997) 98–110.
[27] J. Domingo-Ferrer, J. Soria-Comas, Data Anonymization, in: J. Lopez, I. Ray, B. Crispo (Eds.), Risks and Security of Internet and Systems, Springer International Publishing, Cham, 2015, pp. 267–271.
[28] Y. Liang, R. Samavi, Optimization-based k-anonymity algorithms, Computers & Security 93 (2020) 101753.
[29] A. Meyerson, R. Williams, On the complexity of optimal K-anonymity, in: SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2004, p. 223.
[30] K. Doka, M. Xue, D. Tsoumakos, P. Karras, k-Anonymization by Freeform Generalization, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ACM, New York, NY, USA, 2015, pp. 519–530.
[31] K. Guo, Q. Zhang, Fast clustering-based anonymization approaches with time constraints for data streams, Knowledge-Based Systems 46 (2013) 95–108.
[32] J. Cao, B. Carminati, E. Ferrari, K. Lee Tan, CASTLE: A delay-constrained scheme for k-anonymizing data streams, in: 2008 IEEE 24th International Conference on Data Engineering, IEEE, 2008, pp. 1376–1378.
[33] A. Otgonbayar, Z. Pervez, K. Dahal, S. Eager, K-VARP: K-anonymity for varied data streams via partitioning, Information Sciences 467 (2018) 238–255.
[34] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–12.
[35] M. A. Husnoo, A. Anwar, R. K. Chakrabortty, R. Doss, M. J. Ryan, Differential privacy for iot-enabled critical infrastructure: A comprehensive survey, IEEE Access (2021) 1–1.
[36] W. Mahanan, W. A. Chaovalitwongse, J. Natwichai, Data anonymization: a novel optimal k-anonymity algorithm for identical generalization hierarchy data in iot, Service Oriented Computing and Applications 14 (2) (2020) 89–100.
[37] A. V. Vasilakos, Z. Li, G. Simon, W. You, Information centric network: Research challenges and opportunities, Journal of Network and Computer Applications 52 (2015) 1–10.
[38] R. Ravindran, Y. Zhang, L. Grieco, A. Lindgren, J. Burke, B. Ahlgren, A. Azgin, Design Considerations for Applying ICN to IoT, Tech. rep., ICN Research Group (2019).
[39] D. Gupta, S. Rani, S. H. Ahmed, R. Hussain, Caching Policies in NDN-IoT Architecture, in: Integration of WSN and IoT for Smart Cities, Springer, 2020, pp. 43–64.
[40] C. Tschudin, M. Sifalakis, Named functions and cached computations, in: Consumer Communications and Networking Conference (CCNC), IEEE, 2014, pp. 851–857.

[41] M. Król, I. Psaras, Nfaas: Named function as a service, in: ACM ICN, Association for Computing Machinery, 2017, pp. 134–144.
[42] Y. Abidy, B. Saadallahy, A. Lahmadi, O. Festor, Named data aggregation in wireless sensor networks, in: IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–8.
[43] N. Naik, Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http, in: IEEE International Systems Engineering Symposium (ISSE), 2017, pp. 1–7.
[44] C. Akasiadis, V. Pitsilis, C. D. Spyropoulos, A Multi-Protocol IoT Platform Based on Open-Source Frameworks, Sensors 19 (19) (2019) 4217.
[45] P. Colombo, E. Ferrari, Access control enforcement within mqtt-based internet of things ecosystems, in: ACM on Symposium on Access Control Models and Technologies, 2018, pp. 223–234.
[46] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, B. Fang, A Survey on Access Control in the Age of Internet of Things, IEEE Internet of Things Journal 7 (6) (2020) 4682–4696.
[47] H.-C. Chen, Collaboration iot-based rbac with trust evaluation algorithm model for massive iot integrated application, Mobile Networks and Applications 24 (3) (2019) 839–852.
[48] J. Domingo-Ferrer, S. Ricci, J. Soria-Comas, A Methodology to Compare Anonymization Methods Regarding Their Risk-Utility Trade-off, in: International Conference on Modeling Decisions for Artificial Intelligence (MDAI), 2017, pp. 132–143.
[49] S. Boriah, V. Chandola, V. Kumar, Similarity Measures for Categorical Data: A Comparative Evaluation, in: International Conference on Data Mining, Vol. 1, SIAM, 2008, pp. 243–254.
[50] R. M. Karp, Reducibility among combinatorial problems, in: Complexity of computer computations, Springer, 1972, pp. 85–103.
[51] R. Chatterjee, S. Ruj, S. Dasbit, Public Key Infrastructure for Named Data Networks, ACM International Conference Proceeding Series Part F165625 (2020). doi:10.1145/3369740.3369790.
[52] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, M. Wählisch, Information Centric Networking in the IoT: Experiments with NDN in the Wild, in: Proceedings of the 1st international conference on Information-centric networking - INC '14, ACM Press, New York, New York, USA, 2014, pp. 77–86. doi:10.1145/2660129.2660144.
[53] J. Shi, D. Pesavento, L. Benmohamed, NDN-DPDK: NDN Forwarding at 100 Gbps on Commodity Hardware, in: Proceedings of the 7th ACM Conference on Information-Centric Networking, ACM, New York, NY, USA, 2020, pp. 30–40. doi:10.1145/3405656.3418715.
[54] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of SPARQL, ACM Transactions on Database Systems 34 (3) (2009) 1–45. doi:10.1145/1567274.1567278.
URL https://dl.acm.org/doi/10.1145/1567274.1567278
[55] P. Erdös, A. Rényi, On Random Graphs. I, Publicationes Mathematicae 6 (1959) 290–297.