Data Aggregation for Privacy Protection of Data Streams Between Autonomous IoT Networks

Renato Caminha Juaçaba Neto, Pascal Merindol, Fabrice Theoleyre

Data Aggregation for Privacy Protection of Data Streams Between Autonomous IoT Networks

Renato Caminha Juacaba Neto, Pascal Merindol, Fabrice Theoleyre
ICube Laboratory
CNRS / University of Strasbourg
Email: caminha,merindol,theoleyre@unistra.fr

***Abstract*—Many IoT applications rely on data streams, flowing from producers to consumers. Typically, Named Data Networking has been designed to manipulate directly data chunks, and is particularly relevant in IoT networks. However, in multi-tenant networks, privacy is a major concern, and producers may refuse to share the personal data they generate with non trusted stakeholders. To guarantee k-anonymity, producers can require their data to be aggregated with the one of other producers. We propose here a routing scheme based on aggregation, relying on a pub-sub approach. By appropriately constructing and querying the set of offers, i.e. the list of data streams that are collected, aggregated and transformed together, our routing aggregation scheme provides a privacy aware large-scale interconnection, where the consumer does not access directly to individual measurements. Our performance evaluation highlights the flexibility of our solution to accommodate a large set of queries, while still respecting privacy.**

I. Introduction

Applications like smart buildings typically involve a large collection of sensors, possibly in a multi-owner environment. Moreover, several applications may co-exist in the same building [1] or city, e.g., HVAC regulation, fire detection, intrusion detection and so on.

Designing such systems in isolation leads to redundant deployments which are not scalable as multiplying devices increases deployment costs and reduces available radio bandwidth for the network. For instance, occupancy sensors may both be used to reduce heating in unoccupied rooms but also to trigger an intruder alarm if no presence is expected. In complex environment, we may envision splitting large-scale infrastructures into small Internet of Things (IoT) networks that share data with each other according to their own rules.

Adopting a data centric approach is common for such applications [2]. We have three distinct kinds of actors: consumers, producers and brokers, each having possibly orthogonal interests with data at play. Producers create data by sensing the environment, e.g., a smart home sensing temperature indoor, room occupation and luminosity. Brokers are data forwarders interconnecting the two former entities. For instance, an electricity provider may serve as broker between electricity subscribers and big data companies, respecting privacy concerns by anonymizing electricity measurements.

In complex scenarios, brokers will collect data from other brokers, forming large chains of domains between consumers and producers [3]. Smart Cities applications heavily rely on data exchange: a company of taxis will for instance collect the billing information, may use a real-time tracking system to identify the congested areas in the city, and will share its data with local authorities for urban planning [4].

We focus here on data exchanged in the form of data-streams: a consumer subscribes to a flow of information, that may transmitted either periodically or after a given event. Streams include descriptive meta-data, such as geolocation, location type and type of measurement, which consumers can use to construct queries for data of interest [5]. For instance, a big data company may subscribe to the electricity production of smart homes that host a solar panel, to compute real-time statistics for a specific region.

Aggregation functions may aid in saving bandwidth and protecting sensitive data. For instance, averaging a group of values will prevent the reconstruction of original values, guaranteeing the k-anonymity property [6]. In particular, we assume in this paper that each data producer specifies a minimum level of aggregation for the data it produces. It forces brokers to aggregate data-streams from several producers to respect their privacy requirements.

However, in practice, the same data-stream may be advertised via several paths. In a privacy aware framework, the individual measurements are by definition aggregated together, and it must be impossible to aggregate two streams that have data in common. Thus, any forwarding node must be able to detect overlapping streams, and should prevent their aggregation. In this paper, we propose the set of principles and rules to design such a framework: brokers construct and aggregate streams from different producers while respecting both a minimum aggregation requirement and the absence of data overlaps; this leads to a difficult challenge.

The contributions of our paper are threefold:

1) Each router is able to publish aggregated data streams among different IoT domains, creating novel privacy-aware aggregated data-streams;
2) We rely on a pub/sub approach, and thus provide the mechanisms to construct a reply, compliant with the privacy requirements (e.g., k-anonymity);
3) We compare the aggregation capabilities of the proposed scheme with a (naive) unsecured strategy (having no in-network data processing for enabling privacy), that collects data individually from producers and aggregates it at the consumer.

II. Related work

Content Centric Networking approaches [2] are popular in IoT: routers directly manipulate data interests, and adopt a publish-subscribe approach. More recently, Named Function Networking [7] was proposed to directly modify data inside the network. Such paradigm could be extended to address data aggregation in IoT networks.

To efficiently aggregate data from multiple producers and consumers, while maximizing data reusability, Abbasian *et al.* [8] propose to rely on a hierarchical structure. Data is aggregated as close as possible to producers, at the first router in common with two consumers.

Clustering is a popular technique: devices elect another nearby device as cluster head (e.g., based on an energy metric), that will act as a broker [8]. By reducing the number of devices, the authors improve the scalability. Alternatively, we may construct a global spanning tree, but unfortunately, such structures often assume a centralized entity (or at least view), requiring a global knowledge of the whole network [9]. This single point of failure makes this option neither reliable nor efficient enough for large scale deployment.

When data is exchanged in large-scale topologies, privacy represents a major concern. In particular, relying on a cloud infrastructure represents a single point of failure, and pushes the control on an external entity. Homomorphic encryption helps to not disclose private data [10], but is computationally intensive, and some operations still require to download data for local computations. Edge computing helps also to not rely on a cloud [11], but data exchange is still an open challenge in multi-owner topologies, without any direct trust relationship. We aim rather to rely on a processing pipeline of data streams [12], while still guaranteeing privacy.

III. Problem Statement & Objective

We model relations between IoT domains using a directed graph $G(V, E)$, where V denotes the set of vertices (IoT domains), and E the directed edges. An edge $u \rightarrow v$ exists if domain u provides data to its peer v. Some vertices are producers: they accept to send their measurements to their peers (e.g., a group of electricity meters). Inversely, consumers are interested in a set of measurements (e.g., the distribution of electricity production from buildings in a given region). In the middle, peers are brokers, and may forward data they receive from their neighbors.

We model aggregated data as a multiset, denoted $\mathcal{M} = (A, m)$ with A being its set of distinct elements and m the multiplicity function (describing the number of occurrences of each element). In the following, we define the **sum operator to aggregate data**, such that $(A_1, m_1) + (A_2, m_2) = (A', m')$ with $A' = A_1 \cup A_2$ and $\forall x \in A', m'(x) = m_1(x) + m_2(x)$.

Summing data enables *anonymization*. Using a transformation function $f : \mathcal{M} \longrightarrow \mathcal{M}$ allows for reducing data resolution and extracts relevant statistics with in-network processing – increasing network efficiency and data confidentiality by performing data manipulation within the network. We say that f preserves the sum of bags if it holds: $f(f(\mathcal{M}_1) + f(\mathcal{M}_2)) = f(\mathcal{M}_1 + \mathcal{M}_2)$. Thus, whatever the order or the decomposition of a sum, applying f first to the accumulated inputs or not has no incidence on the output when applying f a second time. This means that transformations and aggregation operations can be executed (anywhere in the network and in any order) without changing the final result.

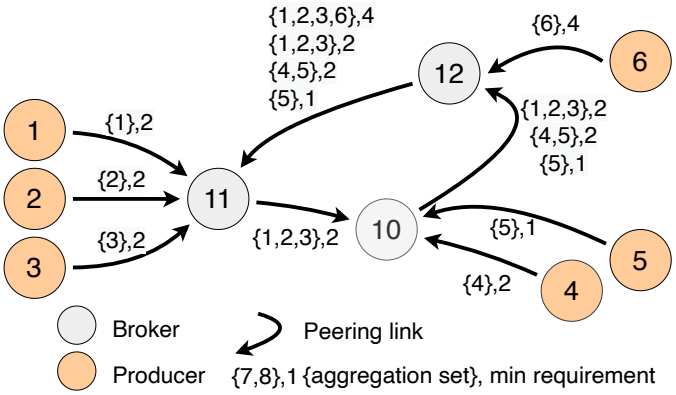

Fig. 1: Announcements of data in a simple network topology.

In this work, we focus on **privacy preserving aggregation functions**. Simplest examples are *min, max* or *average*, that are basic statistical functions where the output, $f((A, m)) = (A', m')$, verifies $|A'| = 1$, i.e. f returns a single value v with $m'(v) = \sum_{\forall x \in A} m(x)$. Note that it also works with any discretization function relying on an arbitrary grain $k \geq 1$ such that $|A'| = k$.

Finally, producers specify its minimum requirement level for its privacy [13]. Thus, the producer can be certain that its individual measurements are not divulged to hosts apart from its direct peers, and enables k-anonymity.

With all these elements, let us now expose the main challenge. Since the topology is meshed, multiple directed paths may exist between any pair of vertices. Thus, a peer may receive the same data through more than one path, leading to intersecting *offers*. Since, by construction, such offers cannot be dis-aggregated, this means that, the same data would be exploited several times in the aggregation function if handled blindly. Thus, our goal is here to propose a routing aggregation scheme able to **efficiently aggregate data offers, without overlaps**.

We adopt a data centric paradigm: producers announce the data they produce, and consumers send interests to receive relevant data [2]. Let us consider figure 1 illustrating the announcement, or the so called offer. We represent producers, generating data, and their peer brokers, forwarding their data after aggregation.

IV. Pub-Sub based Aggregation

Our proposal revolves around connecting producers to consumers via brokers verifying privacy requirements. Each producer is in charge of announcing the streams it aims to export, while brokers re-advertise such data-streams, after

aggregating them with as many other streams as required. We call these advertisement *"offers"*. Such an offer o consists in:

an aggregation set, $AS(o)$**,** that represents the set of producers involved in the offer;

a given requirement level, $r(o)$**,** that represents the minimum level of aggregation required by involved producers. An interest cannot ask for less sub-streams than this value otherwise the specified privacy level is not enforced anymore. It is worth noting that each individual requirement must be verified anywhere in the data stream workflow.

In figure 1 that illustrates these principles, the producer 4 sends an offer o to the broker 10 with the locally produced data, and asks its peer to aggregate its measurement with at least another value ($r(o) \geq 2$) before forwarding its data.

We assume that each producer is uniquely identified by a locally generated *producer ID*. These may be derived from the hash of the network address, and the type of data. For the sake of simplicity, we will use the vertex ID as the resulting producer ID in the rest of the paper.

We also assume that each producer ID is associated with some descriptive metadata. The metadata of a producer p is denoted as $meta(p)$. In the remaining of this paper we will consider that nodes include this metadata in the offers they distribute.

We make a distinction between two kinds of offers: **input offers** correspond to those collected from the direct peers. These input offers are stored in a local *cache* to construct the offers that can be exported (and also used for replying to queries); **output offers** are the input offers that already respect their requirement plus any offer constructed from the input offers. It represents the set of queries it can answer to.

For instance, the input offers $\{\{5\}, 1\}$ and $\{\{4\}, 2\}$ can be merged together to respect the privacy constraint of producer 4. Thus, 10 needs to construct the output offer $\{\{4, 5\}, 2\}$ to respect this requirement. However, the input offer $\{\{5\}, 1\}$ can also be exported alone, unmodified.

Thanks to the indirect identification of producers, in our example, broker 11 is able to detect the potential overlap between the input offers $\{\{1\}, 2\}$ and $\{\{1, 2, 3, 6\}, 4\}$. Indeed, the producer 1 is present in both offers, and the two offers cannot be merged together. Only the second offer respects the privacy requirement, and can be re-exported unmodified. The first offer has to be merged with another input offer to be disseminated.

The number of valid output offers to construct and expose can increase drastically: this combination problem is known to be intractable for large instances. Thus, we propose here an heuristic to define how to construct these output offers from a collection of input offers, that respect the two following properties:

P1: Respect the minimum requirement level of the input offers;

P2: Forbid the aggregation of the same value twice.

Our scheme is based on:

Announcement step where all the nodes construct their output offers, iteratively, from the input offers they receive;

Subscription step where a consumer constructs a query from the set of input offers it received. This query is processed step by step in the network, so that each routing node computes an aggregated value from the value it received from its peers.

A. Announcement step

Discovering data-streams announced by producers along with their requirement levels is the first action that must take place. Initially, each producer is able to create output offers with the locally generated data. It also defines its minimum requirement level for each of its streams.

A node must construct the output offers it will export to its peers. In particular, the node has to respect the properties P1 (minimum privacy level). We make a distinction between:

Complete input offers ($|AS(o)| \geq r(o)$): when an input offer contains an aggregation set that is larger or equal to the min requirement level (i.e., the max among producer requirements involved in $AS(o)$), the node can safely forward the offer as is. Indeed, any consumer receiving this offer can query it as it is and may aggregate it with other non redundant offers if it needs a larger sample of streams;

Incomplete input offers ($|AS(o)| < r(o)$): some producers may send their data with a minimum requirement level strictly larger than the cardinality of their own output offer. While we assume that these producers trust sufficiently their peers to communicate directly their individual data to them, they are not authorized to forward their data without performing the required aggregation (to comply with privacy concerns).

For complete offers, a query may ask any subset of the aggregation set of an offer, provided that the minimum requirement level is respected (the queried subset needs to be larger than this value). An offer is redundant if all the queries it can answer to can already been satisfied by other advertised output offers. For that purpose, we define as $\mathcal{C}_{val}(o)$ all the valid combinations of the aggregation set associated to an offer o: $\mathcal{C}_{val}(o) = \{S \subseteq AS(o) \mid |S| \geq r(o) \vee S = AS(o)\}$. Typically, an offer o_2 is redundant with another offer o_1 if: $\mathcal{C}_{val}(o_1) \subseteq \mathcal{C}_{val}(o_2)$

Algorithm 1 formally describes our proposition to construct valid output offers from input offers. Each node executes this algorithm when it receives novel input offers (either from a peer, or produced locally). Basically, this novel input offer may help to generate one or several output offers (possibly turning some incomplete offers into complete ones), that will be cached in the output offer store.

Algorithm 1 works in the following way:

1) we drop redundant offers when they do not enable new combinations of producers (lines 1-3). These offers are likely to have been propagated via a circuit.
2) the set of available combinations of producers (C^*) is updated from the novel offers (line 4). This set corre-

Algorithm 1: Generating output streams from input streams

Data: Input offers store O^-, Previous output offers O^+
Result: Expanded output offers O^+ if this is a novel offer

1 **foreach** $o \in O^-$ **do**
  // the offer is not redundant
2 **if** $\nexists o' \in O^+ \mid \mathcal{C}_{val}(o) \subseteq \mathcal{C}_{val}(o')$ **then**
3 $O^+ \leftarrow O^+ \cup \{o\}$
// All valid producer combinations
4 $C^* \leftarrow \bigcup_{o \in O^+} \mathcal{C}_{val}(o)$
// Begin expanding offers incomplete offers
5 **foreach** $o \in O^+ \mid r(o) > |AS(o)|$ **do**
  // Find sets of policy-compliant disjoint combinations with the given incomplete offer
6 **foreach** $C' \subseteq C^* \mid AS(o) \in C' \wedge \bigcap C' = \emptyset \wedge |\bigcup C'| > r(o)$ **do**
7 $U \leftarrow \bigcup C'$
  // New offer for disjoint combinations
8 $O^+ \leftarrow O^+ \cup \{(U, |U|)\}$
// Only announce complete offers
9 $O^+ \leftarrow \{o \in O^+ \mid r(o) \leq |AS(o)| \vee o \text{ is local data}\}$
10 **return** O^+

sponds to all combinations of producers available via offers created and received. In practice, computing this set may be computationally expensive but can be reused between executions of the algorithm.

3) it identifies incomplete offers to complete them if possible (lines 5-8). Again, it has to be done without creating any overlap in the aggregation set and has also to satisfy the minimum requirement level.
   Let us consider the conflict graph $CG(V', E')$ where the vertices V' are the available combinations ($V' = C^*$) and an edge exists if they do not intersect ($E' = \{(C, C') | \forall C, C' \in V', C \cap C' = \emptyset\}$). This problem is thus equivalent to finding maximal cliques: all the offers in a clique are pairwise independent. A valid clique contains an aggregated set with a size at least equal to $r(o)$ to respect the minimum requirement level.
   An exhaustive exploration is too expensive for large instances of the problem. Thus, we propose rather an heuristic: we stop the completion as soon as we construct a given number of valid output offers. It is worth noting that we may not succeed to construct a complete offer, because the minimum requirement level cannot be satisfied by construction (e.g., the requirement can be too high according to the underlying graph density). However, when possible, novel input offers received later can help to complete this failed offer.

Algorithm 2: Enabling streams to answer a consumer request

Data: Output offers O^+, Interest $\mathcal{I}(nProd, criteria)$
Result: Producers IDs and set of corresponding combinations to satisfy the interest $\mathcal{I}$, or two empty sets if the interest cannot be satisfied

// Producers that match requested metadata
1 $P_{avail} \leftarrow \bigcup_{o \in O^+} AS(o)$
2 $MP \leftarrow \{p \in P_{avail} \mid criteria \in meta(p)\}$
3 **if** $|MP| < nProd$ **then**
4 **return** $(\emptyset, \emptyset)$ // Not enough producers for $\mathcal{I}$
// All the combinations from available offers
5 $C^* \leftarrow \bigcup_{o \in O^+} \mathcal{C}_{val}(o)$
6 **foreach** $C' \subseteq C^*$ **do**
  // Extract disjoint combinations of producers in MP that match the requirement $\mathcal{I}$
7 $I \leftarrow \bigcap C' \wedge U \leftarrow \bigcup C'$
8 **if** $I = \emptyset \wedge U \subseteq MP \wedge |U| \geq nProd$ **then**
9 **return** (U, C')
10 **return** $(\emptyset, \emptyset)$ // No matching was found

4) all complete offers plus local offers are finally announced to peers (line 9).

In figure 1, node 10 receives a complete offer from 5 (the minimum requirement level equal to 1 is respected). So, it is directly copied as an output offers. However, the offer from 4 is incomplete. Typically, it can be merged with the offer of 5 to form a complete offer ({{4,5}, 2}).

B. Subscription plane

After the offers have been constructed and disseminated after the aggregation (alg. 1), consumers can start subscribing to their interest(s). For this purpose, a consumer needs to select the relevant producers by using their metadata. Then, it must identify a subset of brokers and producers to ask for. More precisely, it constructs a set of input offers that match its query, respecting each requirement level. Finally, a consumer is able to generate a set of interest packets to trigger the subscription: one interest packet per input offer. It is worth noting that each node in the network proceeds in a similar way when it receives an interest.

Replies are constructed and aggregated hop by hop instead of by consumers. Consumers receive the aggregated value, and not the individual measurements.

Algorithm 2 describes how to construct an interest from a collection of input offers:

1) the node identifies all producers (MP) that match the metadata criteria defined in the interest $\mathcal{I}$ (lines 1 and 2);
2) if the number of matching producers is too low, we must discard the interest (line 3) as we are certain that no reply can be generated;

3) we compute all the available combinations of available offers (line 5);
4) we search for a set of combinations that respects the following conditions (line 8):
   a) it results in a valid aggregation by having an empty intersection;
   b) it only uses combinations of producers that match the metadata criteria of $\mathcal{I}$;
   c) the number of producers is enough to satisfy the interest $\mathcal{I}$.

Similarly to the creation of new offers in Algo 1, finding the combination of offers that maximizes the number of producers to answer an interest is intractable for large instances. So, we first select offers that can be used to answer the given request, i.e., $|AS(o) \cap MP| \geq r(o)$ for complete offers and $AS(o) \subseteq MP$ for incomplete offers. Then, instead of checking all combinations, we rather consider it as a starting point and try to merge it with the complement of the others. We finally keep the best combination of producers.

If an interest can be satisfied, the algorithm returns a set of producers that match the interest (U) in addition to the set of combinations to be acquired (C'). Thus, the node can construct one interest per peer (i.e., neighbor), to retrieve the necessary aggregation of data from these producers. Then, each peer will receive a novel (sub-)interest for the combinations found.

Consider that in figure 1, vertex 12 received an interest, and its metadata matches producers $2, 3$ and 4. While consumer 12 can use the offer $(\{1, 2, 3\}, 2)$ received from 10 (asking only for $2, 3$), it cannot use the offer $(\{4, 5\}, 2)$. Indeed, the data produced by 4 must be aggregated with those of 5, but producer 5 does not match the interest. Thus, 4 cannot be retrieved alone.

V. Evaluation

We detail here our performance evaluation: we quantify the efficiency of our solution to aggregate offers that satisfy queries while still respecting the privacy requirements.

A. Evaluation setup

We generate directed graphs with 65 producers and 15 brokers. The broker vertices form a core structure with random edges among them, following the Erdos-Renyi model with edge probability of 30%. Producers connect to the network of brokers via 3 edges to random brokers. We generate graphs in this manner until we reach 30 weakly connected graphs. For the sake of simplicity, we assume that vertices exchange offers synchronously, i.e. instantly at each round.

At each round, we execute algorithm 1 to construct the offers iteratively. After a given number of rounds, we generate 50 interests per consumer. To model the selectivity of the interests (their metadata of interest), we select randomly a subset of the existing producers that match the interest. A random vertex is picked for each interest and this vertex will dispatch it into the network.

We evaluate the ability of our solution to reduce the network load thanks to aggregation [13]. In particular, we rely on the **normalized reply dataset size**, that measures the number of matching producers collectable via available offers. This number is normalized by the number of connected producers, i.e., those which a unaggregated naive collection scheme, i.e. with no privacy, is able to collect. In other words, we evaluate the cost of guaranteeing privacy via aggregation. The unaggregated solution relies on an independent collection of unmodified individual data streams transformed at the consumers.

TABLE I: Evaluation parameters

Global parameters	Value
Broker vertices	15
Producer vertices	65
Edge probability between brokers	30%
Number of brokers per producer	3
Number of graphs generated	30
Number of interests issued in each graph	50
Other parameters for Fig. 2	**Value**
Aggregation requirement	10
Producers requested per interest	40%
Other parameters for Fig. 3	**Value**
Number of message exchange rounds	10
Offer creation limit	10

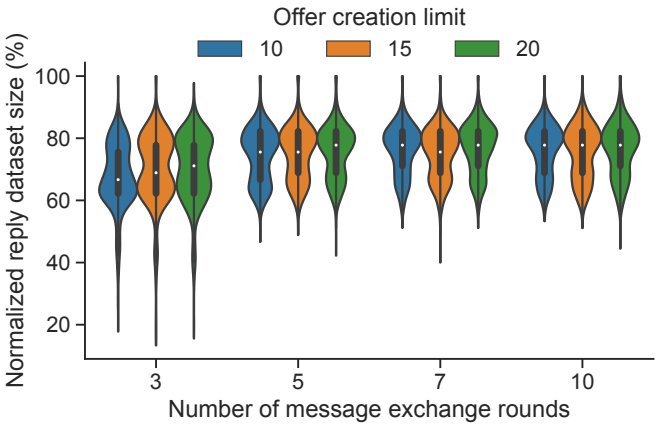

Fig. 2: Conversion analysis

B. Result discussion

We first measure the convergence time of our algorithm by measuring the normalized dataset size after a variable number of rounds (fig. 2). We can note that our scheme converges very quickly: only a few rounds are required to reach stability. We can conclude that creating all possible combinations of novel offers in algorithm 1 is not necessary. It is worth noting that we don't reach 100% since some producers cannot be disseminated in the network because they are in too sparse regions: no vertex can respect the minimum

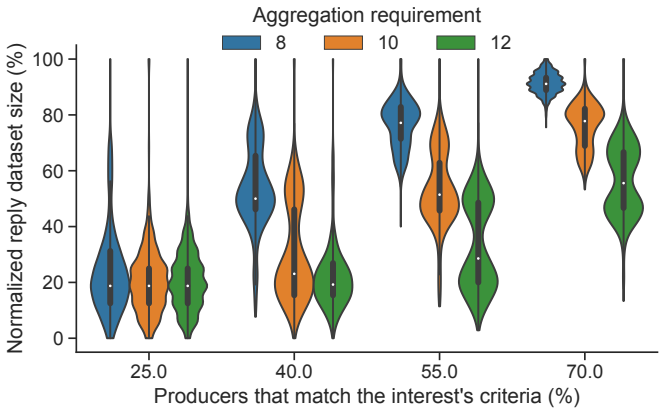

Fig. 3: Aggregation capacity

aggregation requirement, and the metadata of the interests are too selective.

Then, we evaluate the ability of our proposition to deal with complex queries, when only one part of the producers can reply to a query (Figure 3). When most of the producers can reply to a query (that is when a query matches 70% ore more of the overall dataset), and if the aggregation requirement is reasonable (8), close to 100% of such requests can be satisfied regarding a scheme without any privacy. Even with more strict aggregation requirements (10 or even 12), we can retrieve the data from more than 50% of the producers.

We also measure the efficiency when the query becomes more selective: only a subset of producers have matching metadata. Obviously, selective queries are increasing the complexity: the producers are more sparsely distributed in the topology. In particular, the data of some of the producers cannot be aggregated to respect the requirement level. At worst, when the selectivity is equal to 25%, we are only able to use the data generated by 20% of the producers. The more they are matching producers, and the less restrictive the requests are, the more our solution handles the privacy requirements of offers efficiently.

VI. Conclusions

We presented here an adaptative routing scheme exposing and collecting data streams generated by multiple IoT producers. By aggregating different data streams within the network, our scheme guarantees privacy and anonymity. Our proposition relies on a pub-sub paradigm: each device announces the data streams it can export. It piggybacks the unsorted list of related producers, as well as their metadata. We propose a heuristic to expose novel offers that can be generated with those that are received as input. In particular, our announcement scheme constructs aggregations in a distributed manner while avoiding data redundancy. On the other hand, our subscription scheme tries to identify the offers matching a given interest. Our performance evaluation illustrates the efficiency and the limits of our solution to aggregate different offers within the network in order to enable privacy, i.e., consumers do not have direct access to raw individual measurements.

For future work, we expect to propose security mechanisms such that producers can verify that their data is correctly aggregated. Indeed, a consumer may currently attack the infrastructure by generating different interests, with data coming from overlapping producers. Then, the attacker may apply the Gauss-Jordan elimination method to recover individual measurements. Thus, we need a method to detect overlapping interests to keep on providing globally privacy efficient data-streams. We wish to also explore heuristics based on differential privacy [14].

Acknowledgment

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

References

[1] C. K. Metallidou, K. E. Psannis, and E. A. Egyptiadou, "Energy efficiency in smart buildings: Iot approaches," *IEEE Access*, vol. 8, pp. 63 679–63 699, 2020.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, jul 2014.

[3] D. Eckhoff and I. Wagner, "Privacy in the Smart City—Applications, Technologies, Challenges, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 489–516, 2018.

[4] R. Caminha Juaçaba Neto, P. Merindol, A. Gallais, and F. Theoleyre, "Scalability of LPWAN for Smart City Applications," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2021.

[5] P. Desai, A. Sheth, and P. Anantharam, "Semantic Gateway as a Service Architecture for IoT Interoperability," in *IEEE ICMS*, vol. 32, no. 2. IEEE, jun 2015, pp. 313–319.

[6] L. Sweeney, "k-Anonymity: a Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, oct 2002.

[7] M. Król and I. Psaras, "Nfaas: Named function as a service," in *ACM ICN*. Association for Computing Machinery, 2017, pp. 134–144.

[8] S. Abbasian Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, and M. Abbasian Dehkordi, "A survey on data aggregation techniques in IoT sensor networks," *Wireless Networks*, vol. 26, no. 2, pp. 1243–1263, feb 2020.

[9] S. L. Fernando and A. Sebastian, "Iot: Smart homeusing zigbee clustering minimum spanning tree and particle swarm optimization (mst-pso)," *International Journal of Information Technology (IJIT)*, vol. 3, no. 3, 2017.

[10] M. M. Potey, C. Dhote, and D. H. Sharma, "Homomorphic encryption for security of cloud data," *Procedia Computer Science*, vol. 79, pp. 175–181, 2016.

[11] J. Zhang, Y. Zhao, J. Wu, and B. Chen, "LVPDA: A Lightweight and Verifiable Privacy-Preserving Data Aggregation Scheme for Edge-Enabled IoT," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4016–4027, 2020.

[12] E. G. Renart, J. Diaz-Montes, and M. Parashar, "Data-driven stream processing at the edge," in *IEEE ICFEC*, 2017, pp. 31–40.

[13] R. J. Neto, P. Merindol, and F. Theoleyre, "Transformation based routing overlay for privacy and reusability in multi-domain iot," in *NCA*. IEEE, 2020.

[14] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2006, pp. 1–12.