Identifier management in semantic interoperability solutions for IoT

Maria Ganzha*‡, Marcin Paprzycki*§, Wiesław Pawłowski†*, Paweł Szmeja*
Katarzyna Wasielewska*
* Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
† University of Gdańsk, Gdańsk, Poland
‡ Warsaw University of Technology, Warsaw, Poland
§ Warsaw Management Academy, Warsaw, Poland

Abstract—**There are multiple ways, in which large-scale IoT ecosystems can materialize. First, they can be instantiated as a result of a specific project. For instance, when a city authority decides to deploy an IoT-based smart city solution. Second, they can be result of "organic growth". For instance, a port authority deploys smart lighting system, then it implements smart gateways, and in the next stage it decides to combine these two, and a smart container positioning system, into a complete port IoT ecosystem. In the context of interoperability, one of the interesting open issues is: how to manage identities of artifacts and entities that constitute the IoT ecosystem. Identity management seems to be relatively easy in the first case – identifier management can be a part of the IoT deployment. However, as soon as multiple artifacts are to be combined, the problem of identification is no longer easy to solve. This is particularly the case when (i) multiple artifacts originate from different vendors (that use their own identity management approaches), and (ii) when the interoperability solution is based on semantic technologies. In this context, the aim of this contribution is to introduce an approach to identifier management, as well as potential ID interoperability architectures that have been developed within the scope of the INTER-IoT project.**

I. Introduction

By now, it is clear that the vision of large-scale IoT ecosystem is likely to be realized by combining IoT artifacts deployed in "earlier stages" of IoT evolution. Currently, different stakeholders instantiate "local"/"focused" solutions in application domain specific "silos", that are (i) delivered by different vendors (form micro-SMEs to large companies), and (ii) deal with limited in scope problems (e.g. smart lights in a port, smart water metering in a city, local pollution monitoring, etc.). As time passes, such "local" IoT deployments will start to be combined into large(r)-scale solutions, e.g. combining connected (autonomous) cars with a smart city IoT-based parking system. This means an "interoperability solution" has to be employed. As a matter of fact, the EC has recognized this situation and, on January 1, 2016, seven projects devoted to interoperability of IoT platforms have been deployed. One of them is the INTER-IoT project, which utilizes semantic technologies [1], [2], [3] to facilitate communication between artifacts (in particular, concerning interoperability on top levels of the software stack).

One of the interesting problems that materialize both, when a single *new* large-scale IoT deployment takes place, and when multiple *existing* IoT solutions are combined to create large(r)-scale IoT ecosystem is the identifier management. The main issue is to make sure that, in the final IoT ecosystem, each "public" ("visible" to others) artifact has a unique ID. Here, observe that uniqueness of artifact ID is "context-dependent". In other words, if an artifact A has a unique ID *before* joining a "combined ecosystem", *after* joining it, this cannot be guaranteed. It may happen that in the other joining artifacts there are also entities that have the same ID (you can even think about two companies that are subject to a merger, and both of them employ a single person named John Smith and this has to be dealt with in the Human Resources department).

The aim of this contribution is to analyze the problem of identity management and propose a scalable solution. We start with brief outline of INTER-IoT approach to interoperability (Section II). Next, in Section III we introduce typical issues related to IoT identifiers. We follow, in Section IV, with outline of proposed IoT architectures and (in Section V) with discussion of issues materializing in the context of interoperability, during identity roaming and sharing. We complete the paper with discussion of issues specific to the identity management in the INTER-IoT project (Section VI) and conclusions and future work.

II. INTER-IoT approach to interoperability

The INTER-IoT project takes a comprehensive approach to the IoT platform interoperability problem, by offering a solution across the whole communication/software stack. The architecture of the INTER-IoT approach distinguishes five layers, and provides layer-specific interoperability mechanisms for each of them. Altogether, they form the *inter-layer* interoperability framework. More specifically:

- *device layer* – offers Device to Device Gateway, providing a unified interface for access and control over heterogeneous devices, enabling them to interact with each other,
- *networking layer* – uses Software Defined Networking (SDN) and Software Defined Radio (SDR) paradigms, to manage information flow and routing in a heterogeneous networking environment,
- *middleware layer* – achieves interoperability through the INTER-MW (INTER-Middleware) abstraction layer and

subsequent attachment of all platforms/artifacts to it via *bridges*,
- *application services layer* – applying methods of Flow Based Programming, and utilizing service cataloging, discovery, and composition, enables reuse of heterogeneous services offered by artifacts forming the ecosystem,
- *data & semantics layer* – offers common understanding and seamless semantic translation of data and information via the Inter-Platform Sematic Mediator (IPSM) [7], [8].

Here, we are mostly interested in the higher-level layers, especially *middleware* and *data & semantics*, since they explicitly utilize *semantic* methods and thus, require a comprehensive identifier handling solution. Obviously, semantics necessarily needs to describe/refer to (at least provisionally) lower-layer entities, such as devices or services.

At the very heart of the INTER-MW solution lies the message exchange and translation (via IPSM). Internally to the INTER-MW, all messages are represented as RDF [9] graphs, serialized in JSON-LD [10].

Note that, here, we address a very specific problem of identification of entities that messages, exchanged in an interoperable IoT ecosystem, are about. The architectures discussed herein do not offer interoperability solution as such, but should rather be considered as an important part of such a solution. We invite interested readers to study our work concerning syntactic [11], and semantic [8], [12], [13] interoperability, all focused on IoT.

III. IoT ARTIFACTS AND THEIR IDENTIFIERS

Identifiers play an important role in large complex systems (distributed ones, in particular). They may be based on some inherent traits/patterns present in the object to be identified (e.g. DNA profile, fingerprints, etc.), or "artificially" added/created (serial numbers, TAX IDs, car VIN numbers, etc.). Identifiers are particularly important in the World Wide Web, where they are used, in the form of URLs, to refer to almost any kind of resource.

The most important property of any identifier is its *uniqueness* – understood as ensuring that, in a given *context*, there should be only a *single entity having given ID*. Often, other properties, like *persistence* (long-term existence of an identifier) or *immutability* (the same identifier being "always" associated with the same entity) are also considered/required. ISBN, ISSN, DOI, PURL are all examples of persistent (and immutable) identifiers. Some forms of identifiers can also be *actionable* (allowing some action/behavior to be associated with them). Archetypal examples of actionable identifiers are URLs, other instances include DOIs, PURLs, etc.

Obviously, context, in which identifiers are considered, is very important. The larger and more complex the system, the more difficult it is to create and manage identifiers, and achieve their desired properties. To help maintaining the uniqueness, for example, identifiers often take a structural form, which follows certain "hierarchical" or "name-spacing" rules.

In the case of IoT ecosystems, which consist of vast numbers of artifacts, the identifier handling problem becomes critical. Moreover, if one is to pursue the semantic web approach to facilitating interoperability, the omnipresence (and importance) of identifiers still increases – they can point to arbitrary things, their properties, data they produce, populate messages they exchange, etc.

Hence, while identifier management presents a complicated task for a *single* IoT platform, the situation becomes much more complex in the context of *interoperability* of multiple artifacts, forming an IoT ecosystem. It should be obvious that any IoT *interoperability solution* needs to offer methods for managing identifiers for all "public" artifacts involved in the ecosystem, formation of which it facilitated. Since IoT ecosystems are (will be) vastly heterogeneous, it would be totally unrealistic to assume that their constitutive artifacts have (as their natural "property") *globally unique* identifiers. Here, *globally* would mean that if two IoT artifacts join an interoperable IoT ecosystem, all entities that they "bring on board" have identifiers that are *unique* within the formed (joint) solution. Furthermore, there are no guarantees that global uniqueness would be sustained some time in the future, when additional artifacts were included into the ecosystem. Therefore, any identifier management solution that is to work in this scenario, needs to provide means of *forming* unique "global ecosystem identifiers" and mapping them to the "local", artifact/platform-specific ones (and back).

The process of transforming a *local*, platform/artifact-specific, identifier into a *global* one can build upon the original identifier, but it can also provide/generate a completely fresh, "synthetic" ID. In each case, correspondence between both identifiers should be *consistent*, i.e. the interoperability solution should guarantee that the matching of identifiers is deterministic, and (optionally) persisted and recalled, when needed. Furthermore, certain additional "technical" considerations may need to be taken into account. For example, it might be necessary to ensure a specific syntactic form of the formed global identifier, which might include *hashing*, *encoding*, etc.

In the context of an interoperability, each artifact to be added to an ecosystem may have its own "identifier system", which includes the format, syntax, and semantics of identifiers, e.g. allowed characters, maximum/minimum length, namespacing, identifier composition and hierarchical sub-structures (e.g. a part of an ID string that is dedicated for an IoT platform name + device name), etc. Such systems can vary greatly, and it should be assumed that they are not directly compatible, i.e. IDs established within one artifact cannot be used in another. We call those *local ID systems*. Therefore, according to the "common centerpiece" principle of interoperability (reduction to a single common standard), we have to develop a *global ID system*, which, in theory, should be able to accommodate identification of any entity in/from artifacts participating in the ecosystem.

Because we work under the assumption of incompatibility between *local ID system*s, a "naive" *global ID system* that would simply collect and accept any ID from participating artifacts, is not feasible, and, more importantly, does not solve the problem at hand. Instead, we consider the *global ID system* *that has two core capabilities*: (i) to bring all identifiers under a common format, syntax, and semantics; and (ii) to be able to identify entities within local ID systems, from which they originate. To this end, an *ID mapping* solution is required.

ID mapping is to be realized by a software component called *ID mapper*. Its role is to perform two-way transformations between platform local IDs, and global IDs. The transformations are mappings, where a single item in the map is a pair comprised of a local ID, and the corresponding global ID. In broader terms, any such transformation can be understood as a pair of functions, one mapping local IDs to global ones, and the other defining the opposite translation. Typically, these functions will be "one-to-one", although, in some cases, this assumption might be relaxed at the cost of introducing a suitable "disambiguation mechanisms" [14]. In what follows, we shall assume they are one-to-one, and further develop the idea of ID mappers into concrete architectures.

IV. Identifier management architectures

Let us now describe, and critically analyze two generic ID Mapping architectures. Architectures are presented in the context of an IoT interoperability system that involves IoT *platforms*, and an *interoperability application*, which offers some services, and to which the platforms connect, in order to either(i) use the services directly, or to (ii) use the application to intermediate in communication between platforms.

A. Local ID Mappers

The first architecture (depicted in Figure 1) assumes existence of local ID mappers. Each such mapper is dedicated to a platform, and its local ID system. In order to enable interoperability, generated mapping must conform to a number of conditions, which are centered around uniqueness. Local uniqueness of the mapping means that for any local, or global identifier, there exists only one mapping pair. Global uniqueness requires that all mappings that use a given global identifier (which includes mappings in all local ID mappers in a system), refer to a single entity. Both uniqueness constraints are further explained in what follows, and touched upon in Section V, where we present a way to relax them.

The persistent ID mapper storage helps in satisfying the uniqueness constraints, as well as other challenges of ID mapping. We assume, that a storage is initialized empty, and it is filled with mappings during the course of system operation. Explicitly, this means that the ID mapper does not have "full knowledge" about entities in the system (neither within its own platform, other platforms, nor the interoperability application). This implies a number of *ID mapping scenarios*, that need to be addressed:

1) When a global ID, that is not in the storage, comes to the mapper, it needs to generate a new unique local ID, and store the mapping.
2) When a local ID, that is not in the storage, comes to the mapper, it needs to generate a new unique global ID, and store the mapping.

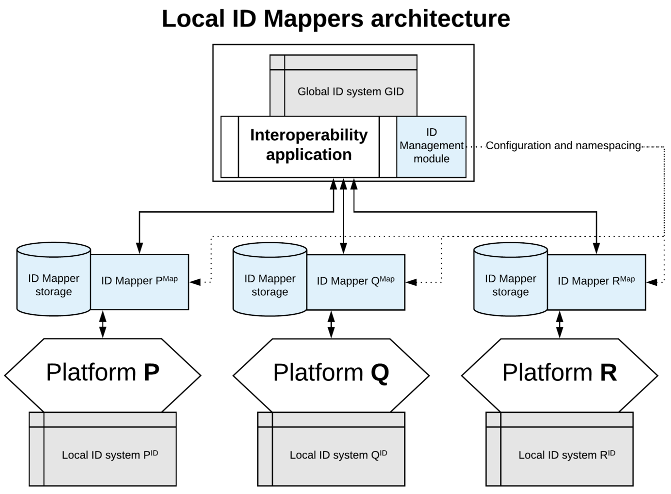

Figure 1. Local ID mapper architecture

3) When either a local ID, or a global ID, that already has a mapping in the storage comes to the mapper, it must use the existing mapping.

Whereas the third scenario can be considered the "usual operation of a mapper", the first two can be potentially non-trivial. The first scenario necessitates that the Local ID mappers must be familiar with the restrictions and intricacies of their local ID system. Any ID that they put out towards the platform, must be compliant with the local ID system. Moreover, it must be able to make (i.e. generate) new local IDs, and decide, whether they are unique, in the scope of the platform. Local uniqueness is satisfied, once those conditions are met. There are many ways to implement the generation of new IDs, some of which include asking the platform itself, to create and identify a new virtual entity, or simply to tell, whether a local ID is already taken. In principle, the "unrecognized" global ID will appear, if a platform receives information about an entity from outside of it. In some interoperability systems, (by design) this will never be the case, hence the local uniqueness problem is simplified. In later sections, we describe the impact of this ID mapping scenario within the INTER-IoT.

The second ID mapping scenario occurs because of our assumption that the mapper does not have full preexisting knowledge about all local IDs. It simply involves making a new global ID from a local ID, and storing the mapping. This, obviously, necessitates familiarity with the global ID system (format, semantics and syntax). To ensure global uniqueness, the new global ID may not be (or have been) used to identify any other entity. This requirement may be solved with various implementations that involve communication with a global ID management module (a part of the interoperability application). similarly to the solutions to the first ID mapping scenario, the local ID mapper may simply ask the interoperability application for a new unique global ID, or instead, just ask to verify if a specific global ID is already taken. Alternatively, the local ID mappers may be configured by the interoperability application, which should provide a mechanism, by which the local ID mapper is able to generate unique global IDs on its own. A simple approach along this line, is to globally manage namespaces, and assign each local mapper a unique namespace, which must be included in the global ID generated by local mappers.

B. *Global ID Mapper*

The second architecture involves a global ID Mapper. Principles behind such mapper, as well as services offered by it are very similar to a local ID mapper, with the important distinctions and differences that stem from the fact that, in our architecture, the global ID mapper would be the only ID mapper in the system. It is also subject to the global, and local uniqueness constraints. The three mapping scenarios identified in subsection IV-A also apply to this architecture.

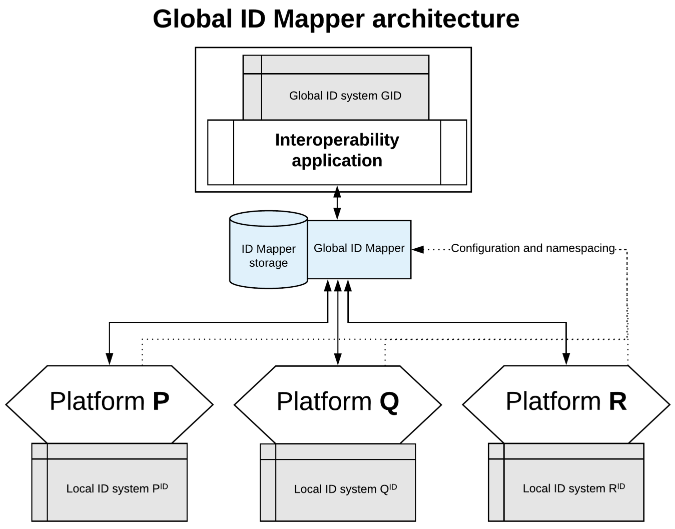

Figure 2. Global ID mapper architecture

Because of the lack of mappers dedicated to each platform, the global mapper needs to be, to some extent, familiar with the constraints placed upon all local ID systems of participating platforms. Because of that, it should be (partially) configurable by the platforms, in order to enable generation of new local IDs acceptable by any platform, to which an identifier is addressed. This requirement stems from the first ID mapping scenario, where a global ID, previously not encountered, appears at the global ID mapper. In this architecture, the only source of such ID can be the interoperability application. If, however, the role of the application is only to manage entities on behalf of the platforms, and serve as a middle man that does not generate any new virtual entities on its own, there will never be an "unfamiliar" global ID. Whether that is the case depends on the specific implementation of the interoperability application.

In any case, if a new global ID is required on the side of the application, it means that (following the first mapping scenario) there is an intention to inform a platform about a new entity. A new local ID must be generated (and persisted as a mapping pair in the mappers storage) for a specific platform. Similarly to the local ID architecture, there are many ways to verify whether a new local ID is valid (i.e. compliant with the local ID system) and locally unique (see Section IV-A).

It is worth mentioning that generation of new local IDs, in the global mapper i.e. outside of the scope of the platform that manages the local ID system, is the most problematic operation that can be programmed in this architecture. However, in a significant number of systems, identifiers will come exclusively from platforms. In those cases the problem of local ID generation in the mapper *will not occur*.

The second ID mapping scenario is easier to manage in this architecture, as opposed to the local ID mappers. Whenever a new local ID appears, it is managed in one central component (the global ID mapper), so the "assignment" of a new global ID can be trivially ensured to be unique by simply checking it against the global IDs already in the mappers storage. Because there is no other mapper storage in this architecture, the uniqueness can be, in this way, easily guaranteed. The process may be somewhat optimized by using namespacing, where each platform has its own namespace used as part of newly generated global IDs, but this is in no way required.

V. Roaming, sharing, and scaling

A number of interesting issues arise when the idea of interoperability is expanded to include management of one entity by multiple platforms (*sharing*) and moving entire management responsibility of an entity from one platform to another (*roaming*). Note that, in this work, we focus exclusively on identifier management, an we do not discuss any other practical and theoretical problems that arise with roaming and sharing, such as data transference and synchronization, data duplication, security (authentication and authorization), etc.

Let us look at roaming first. In terms of identifiers, this means that, from the point of view of the whole system, after the move, a specific entity gets a new identifier, while the old one becomes invalid, i.e. it should no longer be used to identify that entity. However, note that, in terms of ID mappers, the global ID that previously identified the entity should stay the same. This implies that a change in mappings is required, to substitute the old local ID for the new one, in every mapping pair present in any mapping storage.

The global ID mapper architecture enables a relatively easy solution to this problem, because all mapping pairs are stored centrally, and (similarly as with the global uniqueness) we can be sure that we have full access to all mapping pairs. Therefore, substituting old IDs for new ones is, a "simple operation". In this case, platforms that wish to engage in roaming, should inform the global ID mapper, and the interoperability application, about it so that they can update their information. The ID roaming information should simply include the old, and new local IDs.

The local ID mapper architecture responds somewhat less elegantly to the roaming ID problem. In principle the global ID should stay the same, and should be properly "attached" to the new local ID. This requires removing mapping pairs with this global ID from the roaming source mapper, and creating new mapping pair in the target mapper. Because, according to the architecture, the mappers do not communicate directly (nor do they need to know about each others existence), the ID mapping module of the interoperability application must act as an intermediate. While exactly how this operation is implemented can vary, in general, it is a complicated one. In order to perform it correctly, the ID management module must receive the information about the global ID, the old local ID, and a new local ID; and send it to the target ID mapper. Such scheme requires that the platforms first agree on the new ID, in a way that is explicitly known to the "old" platform. The old platform needs to be able to forward this information (ID roaming information) to the ID management module, which must then forward it to the target mapper (so the ID mapper information must also be properly addressed). The target mapper must be enabled to receive the whole packet of information, and properly interpret it, in a sense injecting a new mapping pair (that was authoritatively ordered by the ID mapping module) into its storage. Alternatively, the ID mappers may communicate directly. In any case the implementation may become extra complicated and introduce serious security concerns.

Sharing an entity between two or more platforms means that it is present in all of the sharing platforms, but none of them has the "exclusive rights" to use or manage it. Such, theoretical, situation occurs when an entity is serviced by different platforms, each using it to provide separate, but not necessary different services. For instance, data from an IoT device may be used separately in a visualization service, and an analytics service. When the local ID systems of sharing platforms are incompatible, the entity, in effect, needs more than one local ID (one per ID system).

The local mappers respond very well to the entity sharing problem, even under the uniqueness constraints. Because each storage is effectively per-platform, the uniqueness constraints require only that an entity has one local ID per platform, but does not limit the number of local IDs mapped to a single global ID in the system as a whole. Consequently, sharing does not require any special mechanisms. Different local ID mappers may store mapping pairs for the same global ID, but with different local IDs.

The global ID mapper architecture, as described in Section IV-B, is not prepared to handle sharing. The centralized ID mapper storage is constrained to having only one local ID per global ID, in a mapping pair. This essentially prohibits an entity (uniquely identified by one global ID) to have more than one corresponding local ID. We will revisit this idea in what follows.

In order to support sharing, the global ID mapper architecture must be modified. One such modification is the relaxation of the local uniqueness constraint, i.e. simply allowing multiple local IDs per global ID. However, this solution comes with its own problems; namely, whenever a local ID is communicated towards a platform, the global ID mapper needs to decide, which mapping pair to use (i.e. which local ID should go to that platform). With no additional information, this decision is very hard to make automatically. One solution to that problem, is to segment the global ID mapper storage into partitions, each dedicated to a specific platform. Once each platform has its own set of mappings, the decision which mapping to use becomes very simple, assuming that we know the addressee of an ID. Partitioning the global storage is a substantial change to the architecture, and is, in a sense, introducing the "third" one, a *hybrid*. Here, for obvious reasons, hybrid architecture can be seen as one "between" the global and local ID mapper architectures.

The hybrid architecture, visually similar to the one in Figure 2, offers the easiest solutions to sharing and roaming problems, and is least problematic, when it comes to fulfilling both uniqueness constraints, and realizing the three ID mapping scenarios described in Section IV-A. It is, however, impacted by the problem of generation of new local IDs (a property shared with the "pure" global ID mapper architecture). Another caveat is the single-point-of-failure problem, inherent in any centralized design. The local ID mappers architecture is distributed, and failure of one mapper does not necessarily impact the operation of the whole system. In the global and hybrid architectures there is only one point of failure, and the whole system depends on it. Note that despite this, the global partitioned ID mapper storage present in the hybrid architecture may be implemented as a distributed storage system, and application-independent distributed storage solutions are available on the market. The fact that such software offers data duplication, distribution, but also automatic synchronization is important for the satisfaction of global ID uniqueness constraint.

Another issue, often critical in IoT, is the scalability of the system. The presented architectures are somewhat independent of the scalability problem, because they do not define implementation details. Nevertheless it it worth mentioning that the local ID mapper architecture is, by design, distributed. Local ID mappers are independent and the whole architecture is naturally scalable and easily distributed across processes or machines. The global and hybrid architectures are more centralized, and require specific data storage technologies that offer distribution of both storage and processing of data and database connections. Note that for those architectures the requirement of one common data pool must be satisfied, regardless of whether the data is physically distributed, or not. Services, such as transparent data distribution are offered on lower abstraction layers, than our architectures are about.

One interesting topic, not discussed here in detail, is the combination of multiple ID mapping architecture implementations under one "multi-system". One may consider, for instance, two global ID mappers to be separate ID systems, and apply the same architecture in order to bring them together, treating each, as a new "local" ID system, and mapping it to a new "super-global" ID system. There are many such solutions, each impacting scalability and distribution, and we intend to research them in future works.

VI. Identifiers in INTER-IoT

Let us now look briefly into a slightly more specific situation. In INTER-IoT project, problem of ID management and mapping occurs on the middleware level (introduced in Section II). Various artifacts may want to connect to INTER-MW, and share information about their entities. Each artifact may have potentially different ID system, but the communication syntax is constrained to RDF messages, in a concrete INTER-IoT JSON-LD format. The compliance is assured via the process called syntactic translation, which, simply put, transforms "any format" to RDF. Hence, in this case, identifiers of entities are URIs. Entities in INTER-MW messages, that are exposed to client, use the URI identifiers, that are unique within the scope of a single INTER-MW deployment. As is apparent from the RDF specification, any URI is subject to a set of constraints, which may, or may not be compatible with the local ID systems. To rectify this problem, the GOIoTPex ontology developed within, and for, INTER-IoT semantic interoperability solution, defines a property called *iiot:hasLocalID*. This property can store any textual value, and any entity in INTER-MW messages may be annotated with it. This allows INTER-MW to preserve the local ID of any entity (as a value of an assertion about the *iiot:hasLocalID* property), while using URIs to identify entities both internally, and externally, as the global IDs. The preserved local IDs also serve as information for ID mapping. This is especially important, because in RDF entities may be anonymous, in which case they would not be able to be addressed directly by external artifacts. The *iiot:hasLocalID* property may be mapped to a global ID (URI) in order to a "name" (make non-anonymous) entities.

VII. Conclusions and future work

At the time of writing (January 2018), the INTER-IoT solution to be implemented in INTER-MW (and applied in the two pilot deployments) follows the hybrid architecture. The centralized single-point-of-failure property of the hybrid solution does not introduce architectural changes to INTER-MW. Implementing the local ID mappers architecture would require implementation of many persistent ID mapper storage components "close" to INTER-MW bridges, which are the components responsible for communication with platforms. While it would be much easier to delegate the responsibility of implementation of the mapper to the bridge implementer, it would also introduce the need for bridge persistence, which can otherwise be avoided. The ease of ensuring global and local uniqueness, as well as the lack of additional requirements for the bridges made the hybrid architecture the best choice for INTER-MW.

Obviously, as discussed in Section V, question of scalability remains open. While, conceptually, the hybrid solution should be the best from the scalability perspective in majority of situations, we recognize the fact that this stipulation needs to be thoroughly tested in simulated and real-life deployments. However, this will require not only setting up a testbed, but also developing a comprehensive *set of benchmark-like test scenarios*. Here, we believe that before a credible set of such scenarios can be proposed, knowledge from initial deployments needs to be gathered first. Hence, we plan to come back to this issue by the end of this year.

Acknowledgment

This work was partially supported by the European Union's "Horizon 2020" research and innovation program as part of the "Interoperability of Heterogeneous IoT Platforms" (INTER-IoT) project under Grant Agreement No. 687283.

References

[1] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "Semantic interoperability in the Internet of Things: an overview from the INTER-IoT perspective," *Journal of Network and Computer Applications*, vol. 81, pp. 111–124, March 2017.

[2] ——, "Semantic technologies for the IoT – an Inter-IoT perspective," in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. Berlin, Germany: IEEE, April 2016, pp. 271–276.

[3] ——, "Towards semantic interoperability between Internet of Things platforms," in *Integration, Interconnection, and Interoperability of IoT Systems*, R. Gravina, C. E. Palau, M. Manso, A. Liotta, and G. Fortino, Eds. Springer, 2017, pp. 103–127.

[4] R. Mavlyutov, M. Wylot, and P. Cudré-Mauroux, "A comparison of data structures to manage URIs on the web of data," in *12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 – June 4, 2015. Proceedings*, ser. Lecture Notes in Computer Science, vol. 9088. Springer, 2015, pp. 137–151.

[5] P. Golodoniuc, N. Car, S. Cox, and R. Atkinson, "PID Service – an advanced persistent identifier management service for the Semantic Web," in *21st International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, December 2015. Proceedings*, T. Weber, M. McPhee, and R. Anderssen, Eds., 2015, pp. 767–773.

[6] J. M. Batalla, P. Krawiec, M. Gajewski, and K. Sienkiewicz, "ID layer for Internet of Things based on Name-Oriented Networking," *Journal of Telecommunications and Information Technology*, vol. 2, pp. 40–48, 2013.

[7] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "Alignment-based semantic translation of geospatial data," in *3rd International Conference on Advances in Computing, Communication & Automation (ICACCA), Proceedings*, in press.

[8] ——, "Streaming semantic translations," in *21st International Conference on System Theory, Control and Computing ICSTCC, Proceedings*. IEEE, 2017, pp. 1–8.

[9] "Resource description framework (RDF)," https://www.w3.org/RDF/.

[10] "JSON-LD 1.0. a JSON-based serialization for Linked Data," https://www.w3.org/TR/json-ld/.

[11] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, K. Wasielewska, and C. E. Palau, "From implicit semantics towards ontologies—practical considerations from the INTER-IoT perspective (submitted for publication)," in *Proc. of 1st edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems*, 2017.

[12] P. Szmeja, M. Ganzha, M. Paprzycki, W. Pawłowski, and K. Wasielewska, "Declarative ontology alignment format for semantic translation," in *3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU 2018)*, submitted.

[13] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "Towards common vocabulary for IoT ecosystems—preliminary considerations," in *Intelligent Information and Database Systems, 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3-5, 2017, Proceedings, Part I*, ser. LNCS, vol. 10191. Springer, 2017, pp. 35–45.

[14] A. Jaffri, H. Glaser, and I. Millard, "URI disambiguation in the context of Linked Data." in *Proceedings of the Linked Data on the Web Workshop, Beijing, China, April 22, 2008*, ser. CEUR Workshop Proceedings, C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee, Eds., vol. 369. CEUR-WS.org, 2008.