Streaming Semantic Translations

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

***Abstract*—As the Linked Data paradigm is embraced by a growing number of data providers (and consumers), the count of publicly available, semantically-annotated resources is systematically increasing. Here, elasticity and extensibility of semantics, enmeshed in Linked Data principles, and technologies supporting them, starts to attract application developers. In this context, multitude of developed ontologies motivates (and, in some cases, forces) work devoted to ontology "reconciliation", including alignment and merging, as well as supports pursuing rules of ontology engineering, such as modularity and reuse. At the same time, as uptake of semantic technologies grows, semantically-annotated resources are used not only in static data storage systems (where they appear "in bulk" and are accessed through some query mechanism), but also as separate, dynamically created, transmitted and processed entities.**

In the field of the Internet of Things, this is exemplified in content of messages exchanged, either between internal components of IoT platforms (within the middleware), or between applications built on top of each other. Platforms such as Open-IoT, UniversAAL, or VITAL, use serializations of RDF to transmit messages, while other platforms, such as FIWARE and OneM2M support ontologies, even if they don't use RDF within exchanged messages. However, one of important unsolved issues remains – "how to facilitate understanding of messages exchanged between artifacts founded on different ontologies" (i.e. establish semantic interoperability). In this paper, we focus on semantic translation applied in the context of "continuous" communication between IoT artifacts. Our approach is rooted in semantic translation, based on ontology alignments, and utilizes scalable streaming infrastructure that supports diverse communication topologies and scenarios. It also forms foundation for the design and implementation of the *Inter Platform Semantic Mediator* (IPSM), which is one of key components to establish interoperability in the INTER-IoT project.

I. Introduction

Even a simple reflection on the state of the Internet of Things, A.D. 2017, shows that the IoT ecosystem consists of large (and fast growing) number of separate artifacts – devices, platforms, applications, services, etc. In a typical scenario, such artifacts work in more or less independent silos, while "completing assigned jobs". As a matter of fact, one of the key factors slowing adoption of IoT-based solutions is lack of interoperability. In other words, it is very difficult to facilitate exchange of information between entities developed by separate vendors, deployed using different technologies, "speaking different languages", to achieve independent goals. Here, each artifact (group of artifacts) has implicit, or explicit, semantics – the meaning assigned to its data. Explicit semantics can be represented as an ontology, which provides a comprehensive description of data that includes definitions, categorizations, as well as restrictions put on it, and logical rules it must adhere to. In the case of implicit semantics, it is possible to "make it explicit" (i.e. establish an ontology that represents it) and develop translators between the internal data representation and the ontology-based data representation (for more details, see [1], [2]). Therefore, in what follows, we assume that (one way or another) IoT artifacts have explicit semantics, represented as an ontology.

Let us now assume that a number of IoT artifacts has to be combined to instantiate a new IoT ecosystem. Obviously, it can also be assumed that an existing IoT ecosystem is to be enriched by adding new artifacts (e.g. new sensors/platforms/services) to deliver new functionality to the user(s). To achieve this goal, communication channel(s) has/have to be established to facilitate exchange of information between two, or more, artifacts. While, as assumed, all joining entities have explicit semantics (in the form of an OWL ontology), it may, and in most cases will, differ from entity to entity (see, also [3]). The aim of this paper is to discuss how it is possible to develop an efficient communication infrastructure that will facilitate exchange of messages, combined with their translation "on the fly".

To this effect, we proceed as follows. In Section II we briefly discuss how ontology alignments can be used to translate messages between entities with different semantic representation of the domain of interest. We follow (in Section III), with a description of the alignment format used in our approach. Next, in Section IV, we extend our translation mechanism to the special case of any-to-any translations. This background information allows us to introduce the *Inter Platform Semantic Mediator* (IPSM), a stream-oriented architecture (in Section V). Some special aspects and features of the proposed architecture are briefly discussed in Section VI. The same section also contains conclusions and presents our ideas for future work.

II. From alignments to translation

Let us assume that multiple IoT artifacts, having explicit semantics (represented in the form of OWL ontologies), are to exchange data/information with each-other via, one or more, "communication channels". To make such communication

feasible, a common understanding, i.e. semantic interoperability has to be established (besides syntactic interoperability defining the format of data exchange). In general, there are multiple ways of achieving this goal. The most obvious is to implement the same semantics (the same ontology) across the IoT ecosystem. However, it would mean that each artifact within the ecosystem would, possibly, have to change its original semantics to the common one. For obvious reasons (e.g. who will decide, which semantics should be chosen as the common one, who will pay the cost of such changes, how to make sure that another change will not be necessary when further integration, involving independent IoT artifacts, occurs, etc.) this is impossible to achieve in any real-world scenario. As a matter of fact, our observations are very clear, instead of converging towards a common solution / set of solutions, new IoT platforms materialize. According to O. Vermesan [1], within last year almost 100 new, independently recognizable IoT platforms entered the global market (pushing the total number to about 450).

Seeking a weaker solution, "semi-interoperability" may be achieved when many artifacts write data to the same storage that has a specific semantics. In this case, all data sources need to translate the data on their own, from their own semantics into the selected one. This approach, however, is mainly applicable to data-fusion scenarios, and does not allow for proper inter-artifact communication. Furthermore, the question "which ontology is to be the one selected for the common repository" remains unanswered (and a likely deal-breaker).

Partial interoperability is (at least theoretically) achievable when artifacts share an upper ontology as a module of their ontologies. This level of interoperability, however, is hardly applicable to any realistic case of IoT ecosystems, mainly because of their vast (internal) heterogeneity. To put it simply, in order to be fully functional, the highly specialized applications that are to collaborate within IoT ecosystems, to deliver value to the users, require higher level of interoperability than what, in most realistic cases, an upper ontology can offer. Here, it should be also noted that the heterogeneity of semantic representations occurring in IoT-related domains (see, for instance, [3]) further diminishes real-world applicability of this approach. Consider, for instance, a sample use case, in which an e/m-health-related subsystem is to be used by drivers of a trucking company. Here, the semantic representations of e/m-health domain is to work hand-in-hand with a logistics and transportation one [3].

In the layered model of interoperability [4], semantics is positioned above *syntactic interoperability* [2], which ensures common understanding of the *structure* of data. Therefore, in what follows, we assume that artifacts participating in the ecosystem, have already achieved the syntactic interoperability. Since we also postulated that all of them have defined explicit semantics, we further assume that the information produced/consumed by the artifacts, of the IoT ecosystem, is expressed in RDF [5], the de-facto standard language for the exchange of semantically-enriched data. It should be stressed that the actual RDF serialization (RDF/XML, Turtle, JSON-LD, etc.), used by individual artifacts, is unimportant (does not make any difference in what follows).

In case of IoT ecosystems, we are interested in ensuring that artifacts can use *communication channels* to exchange data between each other, in a meaningful manner. This simple idea turns out to be fairly complex to realize in practice, and requires solutions that harmonize both syntax and semantics of information, besides actually using the communication infrastructure (channels). Note that the actual number and topology of communication channels depends on the information flow in the ecosystem and can change dynamically over time (in response to the needs of the applications using them).

In principle, a *semantic-enabled* communication channel is a medium that, when properly configured, can accept messages with data annotated by entities from one (input/source) ontology, and produce semantically equivalent messages, annotated with another (output/target) ontology. In this way, the translation becomes a part of the communication process and is entirely *external* to the participating artifacts (contrary to the semi-interoperability). Complexity of this task stems from the fact that each artifact might represent a different application domain. However, as stated above (and illustrated in [3]), even within the same domain, ontologies might (and usually do) differ substantially. Henceforth, defining appropriate "translation rules" requires good knowledge of the domains of interest. Additional challenge is to ensure that the communication architecture is capable of handling the volume of messages that can be exchanged between IoT artifacts. Here, it should be noted that the solution proposed here brings about a number of possibilities that lead us to believe that it is likely to scale. However, these considerations are out of scope of current contribution.

Since the conceptualization of a domain directly corresponds to its *ontology* (see, [6]), to bridge the "semantic gap" between the artifacts, we have decided to use "alignments" between the corresponding ontologies (see, [7]). Upon their creation, appropriate alignments can be "applied" to any message traveling through a (semantic-enabled) communication channel. Specifically, application of an alignment substitutes parts of information from the incoming message with their translations, i.e. the descriptions that the specified parts are mapped to, by the alignment. Such, updated/translated, message is then made available to the recipient(s) "at the end of the channel".

The translation process allows, at least in theory, to find an equivalent representation of any piece of data originating from one ontology in terms of another ontology, provided that an appropriate alignment between them exists. If there is no direct equivalence, then one may be able to translate the data to its more general, though less informative, representation in the target ontology. Those two cases (equivalence and lack thereof) directly correspond to two common relations in alignments – *equivalence* and *subsumption*. Note that in-depth treatment of issues related to alignment creation and use (as

[1] O. Vermesan, SINTEF, personal communication, e-mail from July, 2017

well as consequences of subsumption-based translation to the information flow), while being an interesting research topic on its own, is out of scope of current considerations.

In this context, in what follows we shall describe both the mechanisms used for application of alignments, and the communication architecture allowing for efficient handling of messages.

For readability reasons, let us start by focusing on a *one-to-one* interoperability (single semantically-enabled communication channel). The proposed approach will be then extended to the *any-to-any* scenario. To provide foundation of the proposed approach, in the next section, we present the alignment format used for configuring the translation process.

III. Alignment format

Since the semantic interoperability is to be based on application of ontology alignments, from a practical perspective, it is necessary to introduce a way of representing alignments. Therefore, within the INTER-IoT project, we have developed a dedicated format, with an XML representation, based on the Alignment API [8] and inspired, to some extent, by the EDOAL [9]. From here on, we will name this format: the *IPSM format*, since it is to be used in the Inter Platform Semantic Mediator (IPSM) component developed within the scope of the INTER-IoT project.

An alignment, expressed in the IPSM format, consists of a sequence of *cells*, that define parameterized "unit" mappings/transformations applied to RDF data/graphs. The general (meta-level) structure of an alignment in the IPSM format is represented in Listing 1.

```
<Alignment>
<onto1> { source ontology info } </onto1>
<onto2> { target ontology info } </onto2>
<steps>
  <step order="1" cell="cell_k"/>
  { more steps }
</steps>
<map>
  <Cell id="cell_id">
    <entity1> { source RDF pattern } </entity1>
    <entity2> { target RDF pattern } </entity2>
    <transformation>
      { functional constraints }
    </transformation>
    <filters> { datatype constraints } </filters>
    <typings> { typing info } </typings>
  </Cell>
  { more Cells }
</map>
</Alignment>
```

Listing 1. INTER-IoT alignment format — general structure

In principle, Listing 1 describes a uni-directional alignment comprised of independent mapping cells, each of which has an "input" and an "output" entity descriptions. Elements **<onto1>** and **<onto2>** describe the "source" and "target" ontologies of the alignment, by giving their URIs and specifying the formalism used for their definition (e.g., OWL 2.0). Here, let us note that, in the case when bidirectional translations are needed, separate alignments have to be defined for "each direction" (even if the alignment cells all describe equivalences, and are, therefore, trivially reversible). This is because, in the proposed alignment format, source and target ontologies are explicitly specified – this information is necessary for the communication channel configuration/creation process, within the IPSM.

In the Listing 1, the **<steps>** element specifies the (default) **order**, in which *cells* of the alignment will be subsequently *applied* during the message transformation process. Each **<step>** refers to a cell identifier, as given by the **id** attribute of the **<Cell>** element. Note that a given cell might be referenced here (and, hence, also applied) more than once. Furthermore, if needed, the default order may be overridden during the channel configuration process.

Every **<Cell>** element represents an "atomic" transformation applied to the RDF graphs. Here, content of **<entity1>** describes the *source*, and content of **<entity2>** establishes the *target* of the transformation. Both should be valid RDF graphs (presented in the RDF/XML serialization), possibly containing special-purpose nodes, playing the role of "variables", which are to be bound and referenced within the transformation.

Let us now present an example of a specific alignment. The source "pattern" in Listing 2 matches any RDF graph, in which there is a node (matched by **<sripas:node_CTX>**) with two (value) properties: `<wgs84_pos:long>` and `<wgs84_pos:lat>`. Their values will be matched by **<sripas:nod_x>** and **<sripas:node_y>**, respectively.

```
<sripas:node_CTX>
  <wgs84_pos:lat>
    <sripas:node_x/>
  </wgs84_pos:lat>
  <wgs84_pos:long>
    <sripas:node_y/>
  </wgs84_pos:long>
</sripas:node_CTX>
```

Listing 2. Source RDF pattern – **<entity1>**

We can now illustrate the situation when the target entity pattern refers to a value, which has to be "computed" from some of the values matched by the source entity pattern. The target RDF pattern, in Listing 3, refers to the RDF node matched by **<sripas:node_CTX>**, and assigns to it property `<geosparql:asWKT>` with a value referenced by **<sripas:node_z>**. Since **<sripas:node_z>** did not appear in the source pattern, at this point, its meaning is undefined. Here, lack of properties `<wgs84_pos:long>` and `<wgs84_pos:lat>` in **<entity2>** means that the transformation presented by the considered cell *removes* them from the RDF graph matched by **<entity1>**.

```
<sripas:node_CTX>
  <geosparql:asWKT>
    <sripas:node_z/>
  </geosparql:asWKT>
</sripas:node_CTX>
```

Listing 3. Target RDF pattern – **<entity2>**

To give a meaning to the **<sripas:node_z>** element, we need to add some *constraints*, which would define it in terms of the values of **<sripas:node_x>** and

<sripas:node_y>. This is precisely the role of the cell's (optional) **<transformation>** element. The content of this element is a sequence of *functional constraints* (given by **<function>** elements). In Listing 4 we have three such constraints.

```
<function about="str">
  <param order="1"
    about="&sripas;node_x"/>
  <return about="&sripas;node_sx"/>
</function>
<function about="str">
  <param order="1"
    about="&sripas;node_y"/>
  <return about="&sripas;node_sy"/>
</function>
<function about="concat">
  <param order="1"
    val="Point("/>
  <param order="2"
    about="&sripas;node_sx"/>
  <param order="3"
    val=" "/>
  <param order="4"
    about="&sripas;node_sy"/>
  <param order="5"
    val=")"/>
  <return about="&sripas;node_z"/>
</function>
```

Listing 4. Value constraints of the **<transformation>**

Each constraints presented in Listing 4 is of the form:

$$\texttt{fun}(\texttt{arg1}, ..., \texttt{argN}) = \texttt{res}$$

where `fun` is a *function* referenced by the **about** attribute of the **<function>** element, `arg1`, ..., `argN` are *arguments* (described by the **<param>** elements), and `res` is the *result* of applying the function to the arguments (given by the **<return>** element). Both, the arguments, and the result, might refer to the "variable" elements. In this case, the value of the **<sripas:node_z>** element is defined via string concatenation, from the values of **<sripas:node_x>** and **<sripas:node_y>**. More information about functions that can be used within functional constraints will be given in Section V.

The (optional) elements **<filter>** and **<typing>**, in the IPSM alignment format, add *datatype* information to the "variable elements" from the source and target RDF graph patterns, respectively (see Listing 5 and Listing 6).

```
<filter about="&sripas;node_x"
  datatype="&xsd;float"/>
<filter about="&sripas;node_y"
  datatype="&xsd;float"/>
```

Listing 5. Datatype **<filters>**

```
<typing about="&sripas;node_z"
  datatype="&geo-sf;WKTLiteral"/>
```

Listing 6. Data **<typings>**

Translation defined in Listings 2-6, applied to an RDF graph from Listing 7, results in the RDF data presented in Listing 8.

```
@prefix ssn:
  <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix geo:
  <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix xsd:
  <http://www.w3.org/2001/XMLSchema#> .
@prefix p1ont:
  <http://platform1.eu/sensors#> .

p1ont:PositionMeasurementValue
      a ssn:ObservationValue, geo:Point;
geo:lat "52.3"^^xsd:float ;
geo:long "98.2"^^xsd:float .
```

Listing 7. Sample input RDF graph

```
@prefix sosa:
  <http://www.w3.org/ns/sosa/> .
@prefix geo:
  <http://www.opengis.net/ont/geosparql#> .
@prefix geo-sf:
  <http://www.opengis.net/def/sf/> .
@prefix p1ont:
  <http://platform1.eu/sensors#> .

p1ont:PositionMeasurementValue
  a sosa:Result, geo:Geometry;
geo:asWKT "Point(52.3 98.2)"^^geo-sf:wktLiteral .
```

Listing 8. Output RDF after the translation

As stated above, the IPSM alignment format has been developed on the basis of ideas originating from the Alignment API and the EDOAL. Therefore, let us now briefly juxtapose the IPSM alignment format with both of them.

Many existing ontology alignment tools [10] produce their outputs in the Alignment API format, although, mostly, on level 0. This means that only simple class-to-class mappings are expressed (see, [8]). EDOAL, on the other hand, is capable of expressing complex alignments, i.e. mappings between complex descriptions with multiple restrictions. As far as we know, at the time of writing of this contribution (June, 2017), there are no automated tools that produce output in the EDOAL format. Therefore, using EDOAL would, in practice, require anyone to develop her/his own implementation of the language, and accepting all limitations and peculiarities that its authors decided to introduce. Moreover, EDOAL defines its own proprietary set of XML tags, while the cells are not stored in OWL, RDF, or any other well-known ontology language or serialization. Henceforth, we have decided that, at least for the time being, to fulfill our current practical needs, it is better to define our own alignment format, which allows us to express mappings between contents of any valid RDF graphs. At the same time, the original Alignment API format can be easily translated to the IPSM alignment format, because of similarity in their structures. Furthermore, we believe that using RDF/XML for the definition of alignment cells improves readability, understandability and intuitiveness of the format, and, simply, makes it easy to use.

IV. ANY-TO-ANY TRANSLATIONS

Let us now observe that defining alignments, as presented in Section III, allows their application to any message, in order to translate its semantics. Given a pair of IoT artifacts, translating a stream of messages from one to the other requires a single alignment file.

Let us now recall that bidirectional communication requires a pair of "mirrored" alignments, one for each direction of communication. Specifically, while, in theory, reversible alignments are very practical, we realize that it may not always be possible to construct them, especially when subsumption, and not equivalence, is considered. On the other hand, for a large number of alignment cells there exist trivial reverse cells, making construction of a reverse alignment easier. In essence, the problem is reduced to issues that have to be solved for only few specific cells.

For a large number of artifact ontologies, defining all potentially needed *one-to-one* alignment pairs involves a lot of work. Not only it may require aligning every possible pair of ontologies but, also, adding a new ontology to an existing ecosystem grows more difficult, when large number of artifacts are already aligned. Observe that adding artifact N may require creation of as many as $N-1$ *one-to-one* alignments (allowing interoperability with artifacts that are already part of the ecosystem). Instead of doing this, we assume existence of a modular *central ontology* – as described in [1]. Here, the modularity assumption makes the central ontology less vulnerable to changes [11]. In this case, the translation has an "intermediate point", i.e. information is first translated "to the central ontology", and then to the target ontology (with the process being reversed for bidirectional translations). Observe that, the problem of incorporating a new artifact to an existing "federation" is simplified to creation of a pair of alignments between the artifact's ontology, and the central ontology. Bidirectional communication between artifacts, in this case, requires four alignments, i.e. two pairs, one per artifact.

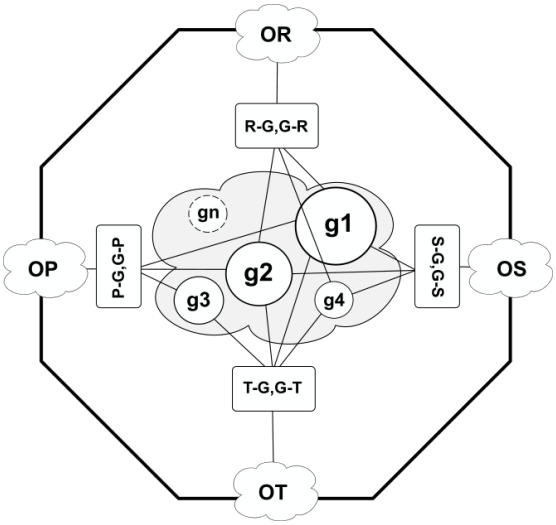

Figure 1. Modularized central ontology and alignments.

Conceptual depiction of use of the central ontology is presented in Figure 1. Here, we assume that, among modules g1, ... , gn, of the central ontology G, some *core modules* model knowledge common to all "participants of a given ecosystem" (artifacts P, R, S, T, with their respective ontologies OP, OR, OS, and OT). For the case of the Internet of Things, a base ontology of the IoT (similar, for instance, to [12], [13]) is a perfect candidate for such a core module (for a more comprehensive discussion of potential candidates for the central IoT ontology, see [14]). In addition to the core modules, the central ontology would contain ontologies/modules that are context/domain specific (e.g. a transport and logistics ontology or the ontology of e/m-healt, or both, see [1], [3]). Each of these modules might, of course, be modular itself.

In order to be useful for translation, the central ontology must cover as much of the range of information expressible by any participant, as possible. Small number of artifacts, especially when they are all known in advance, suggests either using the most expressive ontology, from those used by participants, or making a "super"-ontology encompassing all participants. In such cases the modularity assumption may be relaxed. However, modularity of the central ontology has an important practical consequence (and advantage). An artifact, willing to join the ecosystem, needs to provide only alignments to/from particular modules of the central ontology (depending on the flow of information defined by the application that is going to be developed that involves the joining artifact and the ones already existing in the ecosystem). For instance, in Figure 1, modules g1 and g2 are used by all platforms, i.e. they contain general concepts such as IoT devices, or basic truck identification descriptions. Module g3 models information specific to P and T, while module g4 is pertinent to communication between platforms R, S, T. Therefore, for instance, there will be alignments between ontology of P and module g3, but no alignments between module g3 and R, or S will be needed.

The underlying principle states that an artifact needs only to align the concepts, entities and descriptions, that it wants to use in communication with other artifacts. For instance, in a port environment, a trailer truck company may only want to share the geopositioning information with the port and terminal operators. However, it may not necessarily want to share its catalog of goods. Therefore, in the context of this communication, it only needs to define alignments for the geopositioning module of the central ontology. If, in the future, the platform owner decides to communicate other information with the port, (s)he only needs to define additional alignments for other (appropriately selected) modules, while the already existing alignments may remain unchanged. In this way, a well-established modular ontology defines clear borders between its modules, which enables easy implementation of the described scenario.

Let us now assume that, for each artifact (that has its semantics lifted to an OWL ontology) that is to communicate within a given IoT ecosystem, for all needed / defined communication flows, we have defined all necessary one-way and bidirectional alignments. On the basis of this assumption, we will describe technological foundations and the architecture of the IPSM.

V. Stream-oriented architecture

As eluded above, the *Inter Platform Semantic Mediator* (IPSM) is being developed within the "Horizon 2020" research project INTER-IoT, as a general purpose component for semantics-based translation. The translation infrastructure of the IPSM is built around the notion of a *stream*. Streams, as a data flow abstractions, are certainly not new, but only recently they became widely used and studied. The growing interest in studying them, and their processing, is undoubtedly related to the huge amounts of (real-time) data (or "data in motion", as some call it) sources available today. The need for efficient processing of data streams is, of course, particularly evident in the realm of the Internet of Things.

One of the most difficult aspects of stream processing is the ability to treat data flow in an *asynchronous* and *non-blocking* fashion. Whereas, it's relatively simple to treat any of the two aspects separately, taking them into account together is not easy. Fortunately, ideas originating from the field of *reactive programming* [15] and, recently, codified within the "Reactive Manifesto" [16] led to creation of the *Reactive Streams* specification [17]. Ever since, the number of tools and libraries following this specification is constantly growing. This leads, in turn, to further increase of the interest in both research and practice of reactive stream programming.

When choosing an optimal environment for implementation of the IPSM, we took several options into account. However, eventually, we have decided to use Akka [18] running on the JVM [19]. According to the authors of Akka, it "is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications for Java and Scala" (actually, it is also available for the .NET platform, which makes it "universally available" across majority of most popular software environments of today).

Akka is based on the *actor programming* paradigm [20], [21], and offers a very interesting and useful unification of thread-based and event-based approaches to concurrency. What was also an important factor for the IPSM implementation, was that *Akka Streams* (one of Akka core modules) is a mature and solid implementation of the Reactive Streams specification, additionally offering flexible DSLs (Domain Specific Languages) for building stream-oriented applications.

An overview of the architecture of the IPSM is depicted in Figure 2. The *REST manager* is the "configuration entry-point" of the IPSM, offering two kinds of services, related to the management of: *alignments* and *translation channels*, respectively. The *Alignment repository* is a rather standard service, performing actions related to storing, retrieving and deleting alignments.

More interesting is the *Channel manager*. It offers services for configuring and creating *semantic translation channels*. Because of the "modular central ontology" approach, used by the IPSM, (as discussed in the previous Section), to configure a single translation channel one needs two alignments – having the central ontology as the "mediating point". Note that, creation of bidirectional translation requires instantiation of

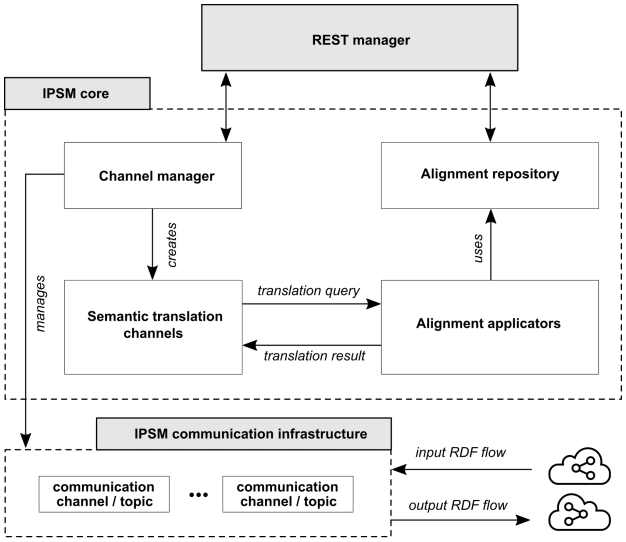

Figure 2. Overview of the IPSM architecture.

two (separate) channels – one in each direction of information flow.

The alignments are subsequently compiled into *alignment applicators*, which are responsible for actually performing the translation tasks. The applicators utilize, appropriately generated, SPARQL 1.1 [22] update queries for every alignment cell. A sample query, being the compilation result for the sample alignment cell, presented in the previous section (see, Listings 2-4), is given (without the CONTEXT declarations) in Listing 9.

```
DELETE {
  ?node_CTX wgs84_pos:long ?node_y.
  ?node_CTX wgs84_pos:lat ?node_x.
} INSERT {
  ?node_CTX geosparql:asWKT ?node_z_typed.
} WHERE {
  ?node_CTX wgs84_pos:long ?node_y.
  ?node_CTX wgs84_pos:lat ?node_x.
  FILTER(datatype(?node_x) = xsd:float)
  FILTER(datatype(?node_y) = xsd:float)
  BIND(str(?node_x) AS ?node_sx)
  BIND(str(?node_y) AS ?node_sy)
  BIND(
    concat("Point(", ?node_sx, " ", ?node_sy, ")")
    AS ?node_z
  )
  BIND(
    strdt(
      str(?node_z),
      <http://www.opengis.net/def/sf/WKTLiteral>
    ) AS ?node_z_typed
  )
}
```

Listing 9. Alignment cell compiled into SPARQL update query

The transformation sections, of the alignment cells, can refer to arbitrary SPARQL 1.1 functions [22] (such as `str` and `concat`, presented in Listing 4). It can also apply user-defined functions, thanks to the efficient *Apache Jena* [23] ARQ SPARQL processor, used by the IPSM. ARQ is the tool directly responsible for transforming the RDF graphs, of

the messages flowing through semantic translation channels. Cell filters, and typings (Listings 5 and 6), correspond to the appropriate SPARQL filters and BIND-ings in the WHERE clause of the update query.

Each translation channel is an Akka Streams *flow*, compatible with the Reactive Streams standard. For connecting the channel to the "outside world" the *IPSM communication infrastructure* uses the *Apache Kafka* [24], [25] message broker. To integrate the IPSM with this high throughput tool, the *Akka Streams Kafka* module is employed. This guarantees that the proposed solution is, again, Reactive Streams compliant. In the process of configuring a translation channel, in addition to the alignments, its *source* and *target* Apache Kafka *topics* have to be provided. Observe that the proposed approach meas that the communication mechanism is based on the subscription mechanism (offered by Kafka). In other words, the receiving party subscribes to a channel where the messages "of interest" (i.e. results of semantic translation based on alignments, specific for that channel) may appear. This allows, among others, very simple realization of one-to-many communication, where all artifacts "interested" in a specific communication subscribe to a given channel. While scalability of the proposed solution was not considered, it is worthy noting that the proposed mechanisms have been build with high throughput in mind. Hence, it is possible to "parallelize" streams to take advantage of modern multi-core and multi-threaded CPU architectures. Experimental investigations concerning scalability of the IPSM, as well as its throughput-oriented optimization (matching architectures of currently existing computer hardware) is planned for the near future.

VI. Remarks, conclusions and future work

Although stream-oriented processing of RDF has already been considered in the literature, known research has concentrated mainly on *RDF Stream Processing* (RSP) and *stream reasoning*. The former area focuses primarily on various ways of extending SPARQL for continuous querying [26] and event processing [27], [28], [29], whereas the latter is geared towards ontology evolution and related problems [30].

The approach to semantic-based translation presented in this paper, seems to open a new perspective on utilizing stream-oriented techniques for RDF processing. It is worthy noting that the architecture described above is flexible, very likely highly scalable, and enables building a complicated network of streaming RDF transformations for arbitrary RDF graphs that flow through the infrastructure. Here, the flows of information within the IoT ecosystem would require, only, instantiation of the needed Kafka channels. Hence, the problem of building an interoperable IoT ecosystem would be reduced to development of an efficient communication infrastructure within a distributed system.

In this context, it is worthy noting that, apart from the "primary" translation scenario presented in previous sections, the infrastructure can be, in fact, used also in other ways. For instance, the alignment for one, or more, ontologies may be empty. As a result, the translation process, for the part that uses such an alignment, would effectively be an identity translation, i.e. it would actually not modify the original message at all. This may be used when the input message already has the semantics of the central ontology, or when a particular communication scenario is intended to *not* to "go through" the central ontology.

An important feature of the IPSM is that it, in fact, does not store the actual ontologies, only the necessary alignments. This makes the requirement of creation of an alignment between two ontologies only a methodological issue, and not technical one. The existing infrastructure may be used with alignments that involve many ontologies and are of any size – from small, one cell alignments, to big mappings with hundreds of cells. Since no restriction is placed on the ontologies, the IPSM may also be used as a very broadly applicable tool for transforming RDF graphs. It is only up to the designer of alignments to decide whether the IPSM is used with specific ontologies in mind, or on a "pure RDF level", where the graph structure and the triples themselves are the focus. Such usage, especially in the context of this article and the INTER-IoT project (where the IPSM was originally developed), considerably deviates from the original usage intentions. As such, we acknowledge the wide possibilities, but do not describe them in detail.

The *streaming element* (i.e. a single message) for the IPSM is a *set* of triples. It should be relatively self-contained, but may reference entities from outside data sources (as is natural in the case of Linked Open Data). Furthermore, it does not have to contain all restrictions placed upon its entities, nor does it need to include definitions of used concepts. It should, however, present enough information to be properly recognized in the context of translation. For instance, if the translation is with respect to devices, then it may require explicit annotation of the entity, in the message, with the device type. In other words, the message needs to match the alignment cells.

This, seemingly trivial, observation is of considerable consequence to the usage of OWL ontologies that are rich in logical rules and complicated restrictions that, in practice, demand a reasoner to fully understand and properly use them. In its current state, the IPSM does not offer reasoning services. For instance, matching an alignment cell for a class *device* will not work for a message with entity of class *sensor*, even when it is known that the *sensor* is a subtype of the *device* (specifically, every *sensor* is a *device*). Currently, adding reasoning capabilities to the translation channels is an interesting open research issue. On the one hand, based on the "wisdom of the crowd" it can be expected that it would slow the developed infrastructure immensely. On the other hand, our initial experiments with real-world semantic processing gives us some hope that the reality may not be as pessimistic. Therefore, our future work involves researching methods of realizing that idea and, at the same time, precisely delineating practical limits of its usability.

Since the IPSM is a *standalone component*, it can be deployed in many instances, even within a single IoT ecosystem, in particular, when the artifacts of the ecosystem can be

divided into "communication clusters", members of which do not communicate with artifacts from outside of their cluster. Obviously, multiple IPSM instances (placed on separate hardware nodes) can also be used for load-balancing and for increasing the total message throughput within the ecosystem.

Finally, let us reflect on the fact that the design of a concrete architecture of communication with the IPSM is a very broad topic, and involves many peculiarities. The possibilities are very varied and include design of channels that may be persistent, or dynamically created on a need-be basis, and, subsequently, destroyed. There are also many details about the philosophy of alignment design and usage, such as the size of alignment cells, order of cell application, and, finally, the number of cells in an alignment. These, and other, details of working with the IPSM (including, above mentioned, experimental evaluation and optimization of scalability of the approach) will be subjects of our future work.

Acknowledgment

This research was partially supported by the European Union's "Horizon 2020" research and innovation programme 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, *Towards semantic interoperability between Internet of Things platforms (submitted for publication)*. Springer, 2016.

[2] 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.

[3] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "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.

[4] "Achieving technical interoperability – the ETSI approach," ETSI White Paper No. 3, April 2008. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/IOP%20whitepaper%20Edition%203%20final.pdf

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

[6] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *International journal of human-computer studies*, vol. 43, no. 5, pp. 907–928, 1995.

[7] J. Euzenat and P. Shvaiko, *Ontology Matching*, 2nd ed. Springer, 2013.

[8] J. David, J. Euzenat, F. Scharffe, and C. Trojahn dos Santos, "The Alignment API 4.0," *Semantic Web*, vol. 2, no. 1, pp. 3–10, jan 2011.

[9] "Edoal: Expressive and declarative ontology alignment language," http://alignapi.gforge.inria.fr/edoal.html.

[10] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, K. Wasielewska, and G. Fortino, "Tools for ontology matching—practical considerations from INTER-IoT perspective," in *Proc. of the 8th Int. Conference on Internet and Distributed Computing Systems*, ser. LNCS, vol. 9864. Springer, 2016, pp. 296–307.

[11] H. Stuckenschmidt, C. Parent, and S. Spaccapietra, Eds., *Modular Ontologies. Concepts, Theories and Techniques for Knowledge Modularization*, ser. State-of-the-Art Survey, LLNCS, 2009, vol. 5445.

[12] "SSN Ontology," http://www.w3.org/2005/Incubator/ssn/ssnx/ssn.

[13] "IoT lite Ontology," http://www.w3.org/Submission/iot-lite/.

[14] 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.

[15] E. Bainomugisha, A. L. Carreton, T. v. Cutsem, S. Mostinckx, and W. d. Meuter, "A survey on reactive programming," *ACM Computing Surveys*, vol. 45, no. 4, pp. 52:1–52:34, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2501654.2501666

[16] "Reactive manifesto," http://www.reactivemanifesto.org/, 2014.

[17] "Reactive streams," http://www.reactive-streams.org/.

[18] "Akka," http://akka.io/.

[19] T. Lindholm and F. Yellin, *Java Virtual Machine Specification*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[20] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular ACTOR formalism for artificial intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, ser. IJCAI'73. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1973, pp. 235–245. [Online]. Available: http://dl.acm.org/citation.cfm?id=1624775.1624804

[21] G. A. Agha, *Actors: A Model of Concurrent Computation in Distributed Systems*. Cambridge, MA, USA: MIT Press, 1986.

[22] "SPARQL 1.1 overview," https://www.w3.org/TR/sparql11-overview/.

[23] "Apache jena," https://jena.apache.org/.

[24] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece*, 2011.

[25] "Apache kafka," https://kafka.apache.org/.

[26] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus, "C-SPARQL: a continuous query language for RDF data streams," *Int. J. Semantic Computing*, vol. 4, no. 1, pp. 3–25, 2010. [Online]. Available: https://doi.org/10.1142/S1793351X10000936

[27] M. Rinne, E. Nuutila, and S. Törmä, "Instans: High-performance event processing with standard RDF and SPARQL." in *International Semantic Web Conference (Posters & Demos)*, ser. CEUR Workshop Proceedings, B. Glimm and D. Huynh, Eds., vol. 914. CEUR-WS.org, 2012. [Online]. Available: http://dblp.uni-trier.de/db/conf/semweb/iswc2012p.html#RinneNT12

[28] A. Llaves, J. D. Fernández, and O. Corcho, "Towards efficient processing of rdf data streams: Short paper," in *Proceedings of the 3rd International Conference on Ordering and Reasoning - Volume 1303*, ser. OrdRing'14. Aachen, Germany, Germany: CEUR-WS.org, 2014, pp. 55–60. [Online]. Available: http://dl.acm.org/citation.cfm?id=2878765.2878771

[29] J. Calbimonte and K. Aberer, "Reactive processing of RDF streams of events," in *Proceedings of the 4th International Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2015) Co-located with the 12th Extended Semantic Web Conference (ESWC 2015), Protoroz, Slovenia, May 31, 2015.*, ser. CEUR Workshop Proceedings, M. van Erp, R. Troncy, M. Rospocher, W. R. van Hage, and D. A. Shamma, Eds., vol. 1363. CEUR-WS.org, 2015, pp. 1–11. [Online]. Available: http://ceur-ws.org/Vol-1363/paper_1.pdf

[30] J. Calbimonte, J. Mora, and Ó. Corcho, "Query rewriting in RDF stream processing," in *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, ser. Lecture Notes in Computer Science, H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto, and C. Lange, Eds., vol. 9678. Springer, 2016, pp. 486–502. [Online]. Available: https://doi.org/10.1007/978-3-319-34129-3_30