Computers and Electrical Engineering 000 (2016) 1–13

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

An architecture for aggregating information from distributed data nodes for industrial internet of things

Tao Zhu[a], Sahraoui Dhelim[a], Zhihao Zhou[a], Shunkun Yang[b], Huansheng Ning[a,*]

[a] School of Computer and Communication Engineering, University Of Science & Technology Beijing, Beijing, China
[b] School of Reliability and Systems Engineering, Beihang University, Beijing, China

ARTICLE INFO

Article history:
Received 1 May 2016
Revised 22 August 2016
Accepted 23 August 2016
Available online xxx

Keywords:
Information aggregation
Product tracing
Industrial internet of things
Semantics
Ontology
Internet of things

ABSTRACT

The current Internet of Things (IoT) has made it very convenient to obtain information about a product from a single data node. However, in many industrial applications, information about a single product can be distributed in multiple different data nodes, and aggregating the information from these nodes has become a common task. In this paper, we provide a distributed service-oriented architecture for this task. In this architecture, each manufacturer provides service for their own products, and data nodes keep the information collected by themselves. Semantic technologies are adopted to handle problems of heterogeneity and serve as the foundation to support different applications. Finally, as an example, we illustrate the use of this architecture to solve the problem of product tracing.

1. Introduction

Internet of Things (IoT) covers a bundle of information and communication technologies, such as identification, sensing, communication and so forth. The ambitious vision of IoT is to interconnect any substantial entity in both the real and digital worlds by extending the Internet and using intelligent interfaces such that everything in IoT can communicate, be identified and interact [1,2]. Although this vision is still a long way from being completed, the development thus far has benefited many areas, including various industries such as healthcare services [3,4], supply chains [5,6], transportation [7] and so forth.

IoT enables an entirely new class of services, such as identification and tracking, with which the operation and role of many industrial systems are being transformed [8]. In [8], the authors surveyed some important IoT industrial applications. For example, using IoT, intelligent transportation systems can be created. From the office, transportation authorities can identify and track each vehicle with loads, monitor its movements, predict load conditions and even recommend driving directions to drivers[7]. In the food supply chain, products can be tracked all the way from farms to retailers, which is helpful in terms of food quality and safety management [9,10].

In these applications, a key fundamental functionality of IoT is to provide a product's information when the product's ID is read. The ID is generally written in radio-frequency identification (RFID) tags. To implement this functionality, there are two representative systems: the electronic product code (EPC) system[1] and the ubiquitous ID (uID) system [11]. These two

* Corresponding author.
E-mail addresses: tzhu@ustb.edu.cn (T. Zhu), ninghuansheng@ustb.edu.cn (H. Ning).
[1] EPCglobal: EPCglobal specifications. http://www.epcglobalinc.org/standards/specs/.

http://dx.doi.org/10.1016/j.compeleceng.2016.08.018
0045-7906/

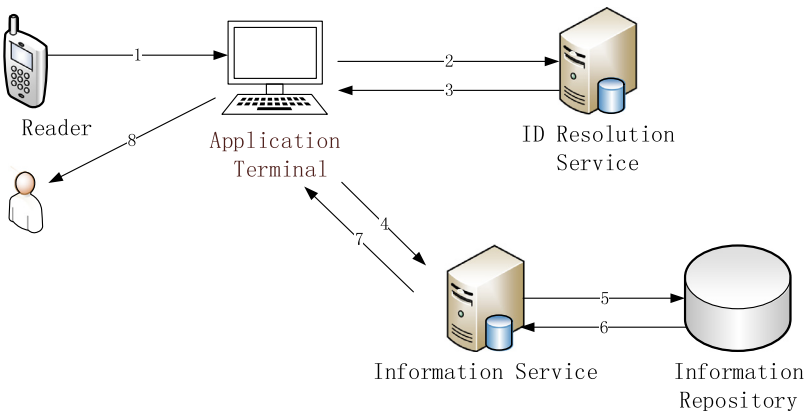

Fig. 1. The classic one-to-one architecture.

systems vary in many aspects, but they have a similar one-to-one (one ID, one record) architecture, as shown in Fig. 1. Once a terminal reads a product's ID, it first forwards it to the ID resolution service, which is called Object Name Service in EPC system and uCode Resolution Server in uID system.[2] The resolution service then returns to the terminal the address of a certain data node that issued the product's ID. This data node generally belongs to the product manufacturer, and we call it the source node of this product. Using this address, the terminal can find the information service provided by the node and obtain the target record from it. Since one ID can be associated to only one record from one data node, we call this a one-to-one architecture.

However, in many applications, a single product could have multiple records spreading over multiple distributed data nodes, and we need all the information to complete some operations [10,12]. For example, in a product supply chain, a product could leave a record at every data node where it arrives and is scanned [13].

The information from all these data nodes is needed to trace the product. The current one-to-one architecture does not directly support such functionality; therefore, supplementary components are indispensable for product tracing. In practice, aggregating information of an object from multiple distributed data nodes is a common task in many industrial applications [10]. Therefore, an architecture that can conveniently aggregate information from multiple distributed data nodes would facilitate many industrial IoT (IIoT) applications, and this is the motivation of our work.

Several requirements should be satisfied for the new architecture to be practical and feasible in multiple applications. First, scalability should be achieved since the development of IoT is very fast. Second, the heterogeneity of information across different applications and data nodes should be considered because we expect the architecture to have a relatively wide applicability. Third, the aggregating operation should clearly be efficient.

In this paper, we propose a new architecture that attempts to satisfy these requirements. The proposed architecture is compatible with the existing IoT infrastructure, including both EPC and uID systems. In this architecture, each manufacturer provides service for their own products, and data nodes retain the information collected by themselves. Semantic technologies are adopted to handle problems of heterogeneity and serve as the foundation to support different applications. To validate the proposed architecture, we apply it to the problem of product tracing.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the preliminaries needed in this paper. Then, in Section 3, we discuss some issues that should be addressed in the information aggregation architecture, which are also the criteria for evaluating an architecture. Section 4 details and discusses the proposed architecture. In Section 5, taking product tracing as an example, we explain how to use the proposed architecture in IIoT applications, in which the use of semantic web technologies is emphasized. Finally, Section 6 concludes the paper and presents directions for future work.

2. Preliminaries

2.1. Semantics for internet of things

Semantic technologies are focused on describing the meaning of information in a formal and machine-processable way, and the resulting descriptions are often called ontologies [14]. Some semantic technologies are used in our work. The Web Ontology Language (OWL) is designed to represent rich and complex knowledge about things, groups of things, and relations between things.[3] SWRL is the Semantic Web Rule Language that combines OWL and RuleML, which is a sublanguage of the

[2] http://www.uidcenter.org.
[3] https://www.w3.org/2001/sw/wiki/OWL.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

Rule Makeup Language designed for formalizing rules.[4] Interested readers are encouraged to refer to related documents presented by W3C.

With rules, SWRL can express facts that hold only under specific conditions; these facts cannot be expressed using OWL alone. An SWRL rule is composed of two parts, antecedent (body) and consequent (head); if the antecedent holds, then the consequent must also hold. Here is an example of SWRL rules: any product whose expiration date is earlier than today is considered to be an expired product.

$$\text{Product(?p1)}\text{\^{}}\text{hasExpirationDate(?p1,?date)}\text{\^{}}\text{SWRLB:greaterThan(today,?date)} \dashrightarrow \text{Expired(?p1)}$$

Currently, there are some reasoners that support SWRL, for example, Pellet from Complexible Inc. [5] Protégé[6] is one of the best ontology editing tools, which can edit ontologies using a sophisticated GUI, and it supports Pellet and other reasoners.

In recent years, there has been an increasing trend of applying semantic technologies in IIoT [15–17]. A main reason for this trend is that the data generated in IoT have the same characters as the Web content. Barnaghi et al. reviewed some scenarios to demonstrate the importance of semantics to the search and development of IoT [15].

- Interoperability: By casting knowledge into unambiguous and machine-processable ontologies, different stakeholders can access and interpret the data unambiguously.
- Integration: By enabling interoperability, semantics technologies can support the seamless integration of data from different sources to create complex abstractions and environments.
- Inference: Semantic web technologies allow logical reasoning, which is able to infer new information or knowledge from existing assertions and rules.

2.2. Product tracing

Product tracing is one of the most important problems in modern supply chains[13]. Product manufacturers would like to know where their products go, and consumers might want to find where a problematic bottle of medicine came from. A supply chain is composed of a number of nodes, including factories, warehouses, and supermarkets. Data are generated when products move from one node to another.

The movement of a product is represented by a series of records distributed on different nodes. The basic schema for a record is (Product, Node, Start, End), which means that a *Product* arrives at and leaves a *Node* at *Start* and *End*, respectively. In [13], the problem of product tracing is defined in SQL language as follows.

Definition 1 (Product Tracing)**.** Given a product o and a time range tstart and tend, tracing o means the following:

SELECT r.Node, r.Start, r.End
FROM Record r
WHERE r. Product $=$ o ORDER BY r.Start AND r.End $\geq$ tstart AND r.Start $\leq$ tend;

3. Information aggregation from federated data nodes in IIoT

From a system-level perspective, the Internet of Things is a radical distributed network system composed of numerous smart objects that produce and consume information [1]. Within this network, a large number of data nodes reside for collecting and storing information from the smart objects, and these data nodes are typically autonomous and could vary in many aspects. Due to this inherent feature of distribution, many applications in IoT often require information that is distributed on multiple data nodes, demanding these data nodes to cooperate and exchange the information to complete tasks.

In Industrial Internet of Things, aggregating information from distributed data nodes is common in many applications. In the healthcare industry, to comprehensively analyse a patient, we need to aggregate information from various IoT-based healthcare service providers that collect data of that patient using various sensors. In the food supply chain, to evaluate food safety, we need information about food production, processing, storage, distribution, and consumption. It is impossible for all this information to be stored in a single data node. Aggregating information from distributed data nodes is the basis of some key enabling technologies. The tractability of objects is key in many applications, such as supply chains, monitoring and surveillance. When the targeted object moves from one location to another, it leaves records collected by different data nodes.

Topology and scalability: As in many other fields, such as federated database systems[18], in IIoT, the architectures for aggregating information from distributed data nodes can be classified into the centralized, the distributed and the P2P architectures [6,19]. The centralized architecture is by far the most impractical one because storing huge volumes of IoT data in a single node can lead to serious scalability issues; therefore, this type of architecture is beyond the scope of this paper.

[4] https://www.w3.org/Submission/SWRL/.
[5] https://github.com/complexible/pellet.
[6] http://protege.stanford.edu/support.php.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

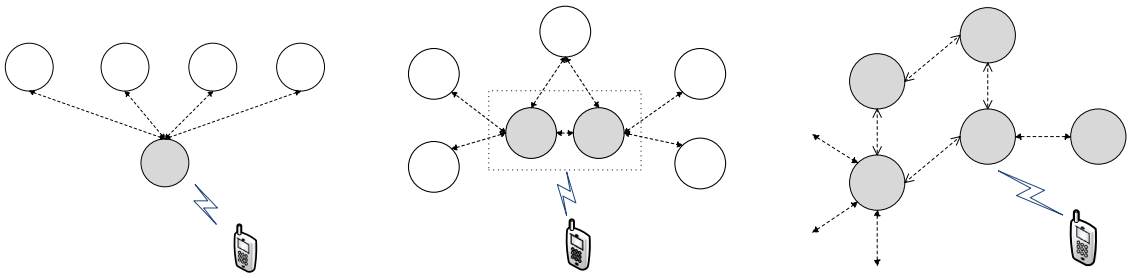

Fig. 2. Architecture taxonomy.

In the remainder of this paper, by centralized architectures, we refer to those architectures with a centralized management node and multiple distributed data nodes, as shown in Fig. 2(a). This centralized management node responds to all requests from all application terminals. However, as the request rate increases, this centralized node becomes the bottleneck of the system. By contrast, distributed architectures distribute the management service among a number of servers, as shown in Fig. 2(b). In this way, the bottleneck problem is relieved. In contrast to both types of architectures, every data node in P2P (also named self-organized) architectures is equipped with a management service to respond to requests for its own information, as shown in Fig. 2(c). There is no global state information that should be shared and coordinated among all nodes, and for a node, it only needs to maintain the connection with a limited number of neighboring nodes [20]. Generally, P2P architectures are considerably more scalable than centralized and distributed architectures [20].

Aggregation efficiency: Aggregation efficiency is another key issue. Although P2P is great at building large scalable systems, aggregation efficiency is its major drawback. Generally, the local data repository of a node is inadequate for supporting the aggregation requests; consequently, the requests must be forwarded to its neighboring nodes for further processing, the neighboring nodes forward the requests to their neighbors in turn, and the forwarding is recursively repeated until all related information is collected. The count of request forwarding would be large and unpredictable. By contrast, centralized architectures have no request forwarding problem because the sole management node possesses all the required information. For some distributed architectures, they also need a small number of forwarding among the management nodes for some requests, but property distribution schemes could avoid this problem and thus enhance the aggregation efficiency.

Heterogeneity: An inherent feature of IoT is heterogeneity. Distributed data nodes in an industrial application system in IoT could be heterogeneous in structure, scale, database management system and so forth. The service-oriented architecture (SOA) has proved to be a successful solution to handle the problem of structure heterogeneity [8,21]. Partners in an SOA comply with a set of agreed external interfaces and implement a stack of communication protocols.

However, for information aggregation, semantic heterogeneity could be a more complex problem. Data nodes are autonomous, and they can choose their ways to store, interpret, and publish the data that they collected. When nodes differentiate in the meaning and use of data, representation of knowledge, semantic heterogeneity arises [15]. With semantic heterogeneity, a data node cannot understand the meaning of information provided by other nodes. We cannot extract the results from the aggregated data unless the semantic heterogeneity is addressed. Aggregating information from distributed data nodes is a common task among many IIoT applications. Building information aggregation architectures that support different applications is worthwhile but more challenging because we do not want to separate applications but rather wish that information from different applications could be shared with each other.

Availability: Availability is a common criterion for any system.[7] A system should not break down simply because of the failure of a single node. In our case, data nodes are autonomous and heterogeneous. Their reliability should not be assumed. Therefore, the availability of the architecture should not rely on any single node.

4. The proposed information aggregation architecture

4.1. The conceptual design

Aiming to achieve information aggregation concerning a thing from multiple distributed data nodes, we propose a new architecture. Its conceptual design is illustrated in Fig. 3.

Every data node assigns a globally unique URI to each information record that it collects during industrial applications. The form of the URI can be uCode, EPC code or any other type. Of course, resolution servers are able to locate the data node of a record by the record's URI, just as in the EPC/uID systems. This assignment is feasible. For example, uCode can be

[7] https://en.wikipedia.org/wiki/High_availability.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

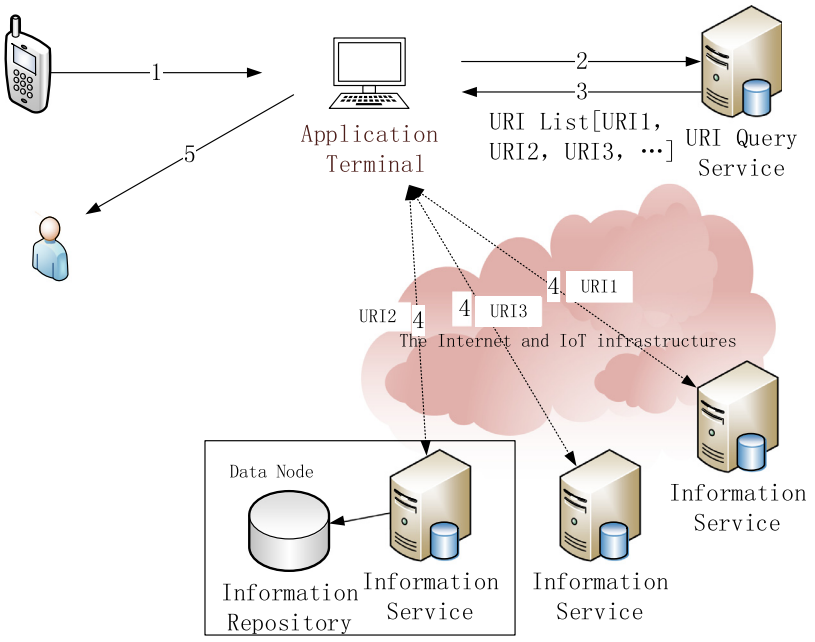

Fig. 3. The conceptual design.

assigned for information and more abstract concepts that do not exist in the real world and for tangible objects and places in the real world.[8]

The key component in Fig. 3 is the query service that provides the UID list of all the information records related to a certain thing on request. For time efficiency, in this architecture, the query service is required to store the UID list in its local data repository. Therefore, it can instantly respond to the requests and does not need to call the other nodes.

With this conceptual design, the information aggregation procedure is divided into the following steps. In step 1, the terminal of a certain IoT application reads a thing's ID. In step 2, a request is forwarded to the query service with the thing's ID, and then the query service responds with a list of the URIs of information records that need to be aggregated. Through the existing IoT infrastructures, it is easy to obtain an information record from a certain data node according to the record's URI. Therefore, in step 3, the terminal concurrently sends requests for all the records that spread on multiple distributed data nodes. In step 4, the aggregated records are integrated and processed, and then they are presented to users.

A major advantage of this conceptual design is the aggregation efficiency. The URI query service maintains the required URIs in its local information repository. To implement this basic design in the proposed architecture, other criteria, in addition to the aggregation efficiency discussed above, should also be addressed, particularly the scalability and heterogeneity.

To make the conceptual design scalable, we need a proper scheme to distribute the URI query service on different nodes. Because the volume of the data and the number of transactions in IIoT have been dramatically increasing, deploying the service on a centralized node would make that node the bottleneck of the system. However, distributing the URI query service on multiple nodes leads to the two following questions: 1) how does a terminal know which nodes to request for a certain thing? and 2) how does a query node know that it should maintain the URIs of which things?

Heterogeneity is the other important problem to be addressed. The difference of data nodes leads to heterogeneity. Data nodes can vary in structure, query language and capacity, among others. Another type of heterogeneity results in semantic difference, when nodes disagree in the meaning and usage of data, representation of knowledge. The problem of heterogeneity becomes more severe when the design is applied across multiple different applications, which means a larger number of data formats, services types and so forth.

In addition to the criteria discussed in Section 3, there is a special issue concerning the URI query service: how can returning unrelated URIs to the terminals be avoided? Terminals of a certain application would not want the information records generated in the other applications.

4.2. The architectures of the components

Considering the above criteria, we proposed the architecture illustrated in Fig. 4. In this architecture, data nodes, query nodes and application terminals communicate with each other via the existing network infrastructures, including the Internet and various IoT systems.

[8] http://www.uidcenter.org/learning-about-ucode.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

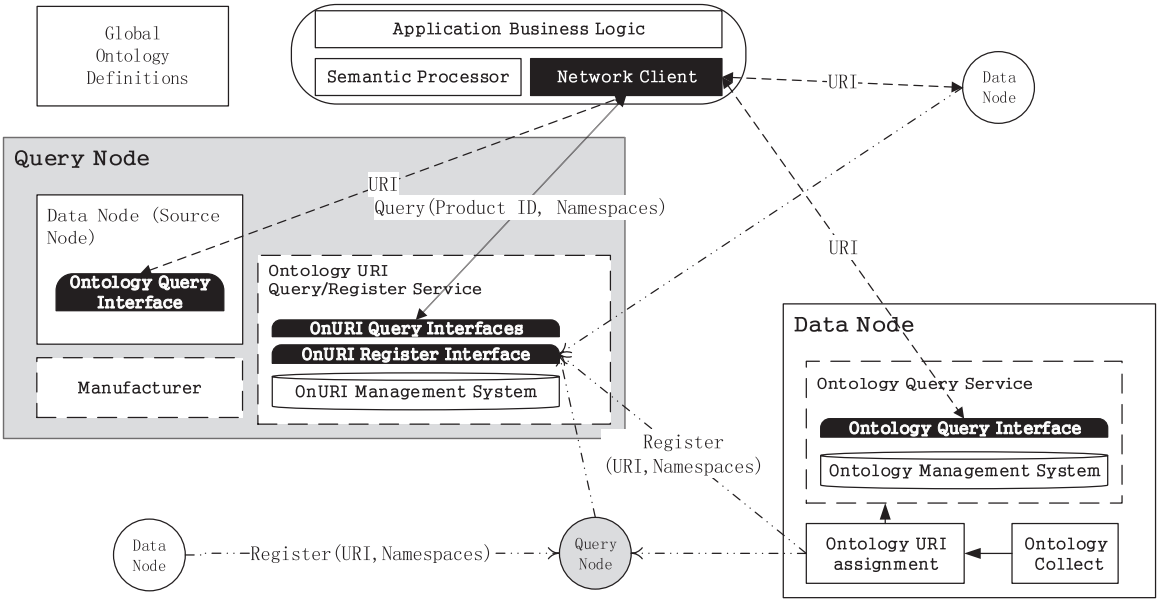

Fig. 4. The architectures of the components.

With the existing network infrastructures, application terminals can quickly locate the source node of a product by its ID. Therefore, in this architecture, the URI query service is distributed on each source node, and each source node maintains the URIs for its own product. With this distribution scheme, for a certain product, application terminals can obtain the address of the query node by the ID resolution services provided by the existing IoT infrastructures.

In our architecture, we leverage the concept of service-oriented architecture and semantic web technologies to handle heterogeneity. SOA has been demonstrated to be successful in integrating heterogeneous systems. Semantic web technologies are useful in handling semantic heterogeneity. Moreover, we can use the reasoning ability of semantic web technologies to complete some tasks rather than developing additional software.

4.2.1. Global ontology definitions

Note that semantics alone are not sufficient to solve the problem of semantic heterogeneity. All the partners involved in the architecture must follow the same ontological definitions. Ontologies from different applications are organized in different namespaces. Of course, some namespaces could be shared among multiple applications.

4.2.2. Data node

A data node is composed of three essential functional components, i.e., ontology collect, ontology URI assignment and ontology query service. Data come from various sources, such as sensors, RFID readers, manually input and so forth. Ontology collect is the component that processes the data and represents data in the form of an ontology. Then, each ontological record is assigned a URI in the ontology URI assignment component. The ontology URI assignment component also analyses what products are involved in each ontological record and then registers the URI at the source nodes of these products. Researchers have developed some systems to effectively store ontologies. For example, Lu et al. proposed a practical system to store ontologies in relational DBMS while simultaneously supporting reason and search [22].

For an ontological record, along with its URI, the namespaces that it used are also sent to the source nodes. The used namespaces are simple but very useful information that indicate what a record is about.

The ontology query service responds to requests from application terminals. This component is similar to the EPCIS in the EPC system and the application information services in the uID system. All the ontology query services of all data nodes provide the same ontology query interface for terminals to follow.

4.2.3. Node

In IoT, each manufacturer is required to provide a data node to maintain the product description of its own products for terminals to query. This data node is the source node of the products. In this architecture, each manufacturer is also required to provide an ontology URI query/register service for the its own products. Each query node is the combination of the data node and the ontology URI query/register service of a manufacturer. All the services of a query node should be accessed by the same IP address. In this way, application terminals can find the address of the service for a certain product through the resolution service of the IoT infrastructures. The ontology URI query/register service has two interfaces for external partners

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

to follow. The OnURI register interface is for data nodes to register the URIs of ontologies about products. The OnURI query interface is for terminals to request the URI list of products. The URI data are maintained in the OnURI management system.

To avoid receiving unrelated URIs, application terminals tell the query node which namespaces they are interested in, and the query node excludes the corresponding URIs that are unrelated with those namespaces. Note that the terminals do not tell the query node about the application that they are interested in. Using namespaces as the selective condition is not only simple but also provides a flexible mechanism for sharing information among different applications.

4.2.4. Terminal

A network client that implements various network communication protocols is necessary for a terminal because there are various types of URIs. In addition, application terminals should be equipped with a semantic processor to support ontology processing and semantic reasoning. Based on the two components, it is the application business logic that implements the specific functionalities that the applications need.

4.3. Evaluations of the proposed architecture

Our proposed architecture is scalable and distributed. The URI register/query service is distributed on all the query nodes (manufacturers). Note that the number of query nodes is not fixed but rather increases with the scale of the system. There are two major merits of this distribution scheme. First, when new manufacturers join the system, although it means more products and more information to be addressed, it does not add additional workload to the existing query nodes. Second, the manufacturers know the scale of their own products; therefore, they could determine how much computational and storage capability the service should be equipped with. Another advantage is that the ontology data are not centrally maintained. Each data node only needs to take responsibility for the data collected by itself and has the ability to selectively share data with other trading partners. Since data are stored in each data node, the workload is naturally distributed.

Manufacturers are willing to provide such a register/query service because this architecture is helpful for them to collect information about their products. In some existing distributed architectures, manufacturers may have to initiate many queries to many nodes to know where the product information is. In contrast, under this architecture, manufacturers preserve all the information URIs of all their products. Furthermore, obtaining these URIs is very easy because they are offered spontaneously by the data nodes. Using data analysis techniques, this product information would provide manufacturers insights regarding many aspects.

The problems of heterogeneity are addressed in this architecture. First, it is a service-oriented architecture. Partners communicate using three agreed upon interfaces, and it does not matter what devices or software that a node adopts. Second, all the records are represented in the form of an ontology following common ontology definitions. Therefore, records can be shared and interpreted unambiguously among all the partners. Even information from different applications could be integrated seamlessly. Moreover, the reasoning ability of semantic technologies is capable of various tasks, and we do not need to develop additional software. For example, in the next section, we will show how to use SWRL to trace products.

The efficiency is another major advantage of this architecture. For any product, the URIs of all the related information have been published to the source node of the product by other data nodes and maintained in a local data repository. Therefore, it can instantly respond to requests from application terminals in real time. In contrast to the P2P architectures, a query may be propagated several times, and this significantly increases the processing time.

The availability of the architecture is supported by its feature of distribution. For centralized architectures, the breakdown of the centralized unit leads to the breakdown of the entire system. For the proposed architecture, the breakdown of any data node or any query node only affects the data or products of that node. However, data availability is the shortcoming. Data are strictly privately kept in each data node. The breakdown of a data node means that the data in this node are not longer available.

One major disadvantage of the architecture is that it could bring heavy network traffic when the ontological records to be transmitted are large. Another disadvantage is that it requires application terminals to possess good processing ability in case the data are too large and to support semantic analysis. Distributed reasoning could be a solution to relieve this problem [23]

5. Case study: product tracing

In this section, taking product tracing as an example, we show how the proposed architecture works for aggregating product information from distributed data nodes in IIoT applications. Note that we have not yet implemented the architecture. The node architectures have been detailed above; thus, we focus on the ontology generation and register procedure and the aggregation procedure. Through this example, we would also like to demonstrate the capacity of semantic web technology in solving problems from industrial applications.

The problem of product tracing has been briefly introduced in the preliminaries section, and an artificial example is illustrated in Fig. 5. In this example, after leaving the manufacturers, a phone and a camera are delivered via some warehouses and finally arrive at the retailer. A special situation occurs during the delivery process in that when the phone passes warehouse2, it is packed in a case with the camera. Now, the retailer employee wants to trace the phone.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

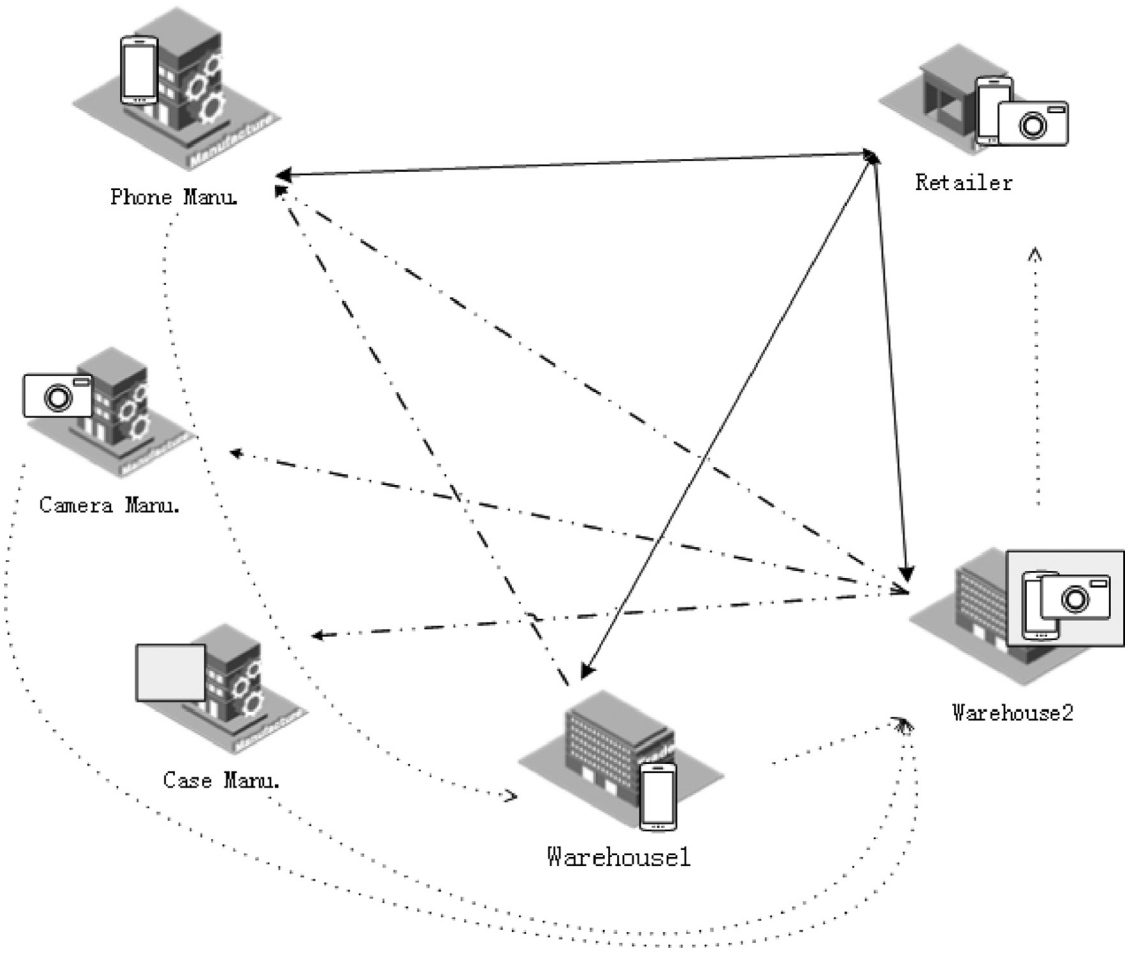

Fig. 5. An example of product tracing.

```
Prefix(:=<http://cybermatics.org/owl/product-tracing/>)
...
Declaration( Class( :TraceRecord ))
Declaration( Class( :Node ))
Declaration( Class( :Product))
Declaration( Class( :PackRecord))
```

Listing 1. Classes declaration.

All the participants in this application use the same ontology definitions. In this case, four main classes are defined in namespace "http://cybermatics.org/owl/product-tracing/ ", denoted as *pt*, to represent product, data node, trace record and the package record; see Listing 1. In this paper, we use SWRL to express the ontology using a functional syntax.[9]

5.1. The ontology generation and register procedure

Within the proposed architecture, data nodes generate information about the phone in the form of an ontology and register them at the corresponding query node, i.e., the phone manufacturer. For this phone, the first and the authoritative piece of information is the product description generated by its manufacturer, which contains some basic information such

[9] https://www.w3.org/TR/owl2-primer/.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

```
Prefix(pt:=<http://cybermatics.org/owl/product−tracing/>)
...
ClassAssertion(pt:Product :phone )
DataPropertyAssertion(pt:hasID :phone ‘‘urn:epc:1:2.24.400’’^^xsd:anyURI)
DataPropertyAssertion( pt:producedBy :phone ‘‘phoneManu’’^^xsd:string)
```

Listing 2. Product description.

```
Prefix(pt:=<http://cybermatics.org/owl/product−tracing/>)
...
ClassAssertion( pt:TraceRecord :phoneManu_record1 )
DataPropertyAssertion( pt:hasProductID :phoneManu_record1 ‘‘urn:epc:1:2.∗.∗’’^^xsd:anyURI)
DataPropertyAssertion( pt:hasNodeName :phoneManu_record1 ‘‘phoneManu.’’)
DataPropertyAssertion( pt:beginTime :phoneManu_record1 ‘‘2016−3−20T21:32:52’’^^xsd:dateTime)
DataPropertyAssertion( pt:endTime :record1 ‘‘2016−3−27T10:22:51’’^^xsd:dateTime)
```

Listing 3. Tracing record.

```
Prefix(pt:=<http://cybermatics.org/owl/product−tracing/>)
...
ClassAssertion( pt:TraceRecord :warehouse1_record1 )
ObjectPropertyAssertion(pt :hasProduct :record2 :phone)
DataPropertyAssertion( pt:beginTime :record1
    201‘‘6−04−01T00:00:00’’^^xsd:dateTime)
DataPropertyAssertion( pt:endTime :record1
    ‘‘2016−04−02T00:00:00’’”^^xsd:dateTime)
...
ClassAssertion( pt:TraceRecord :warehouse1_record2 )
ObjectPropertyAssertion(pt :hasProduct : warehouse1_record2:camera)
DataPropertyAssertion( pt:beginTime : warehouse1_record2
    ‘‘2016−04−01T12:00:00’’^^xsd:dateTime)
DataPropertyAssertion( pt:endTime : warehouse1_record2
    ‘‘2016−04−02T00:00:00’’^^xsd:dateTime)
```

Listing 4. Product tracing.

as the product serial number, manufacturer and so forth. The manufacturer equips the phone with an RFID tag that contains a globally unique EPC, by which the product description can be addressed from elsewhere in IoT. In this sense, this ID is also the URI of the product description. The product description is written in ontology language as follows; see Listing 2.

When the phone leaves the phone manufacturer, it leaves a trace record *phoneManu_record1*, which records the phone ID, thee node name, and time information as follows; see Listing 3.

“pt:beginTime” and “pt:endTime” are data properties of the TraceRecord class indicating when the product is discovered and when it leaves the data node. The manufacturer publishes this piece of information using an HTTP URL and registers this URL along with the used namespaces, including *pt*, at its own ontology URI query/register server.

When the phone passes the next data node, i.e., warehouse1, it leaves the trace record *warehouse1_record1*, as does the camera. Warehouse1 adopts the uID system, so it issues two different ucodes for the two records and registers the ucodes and *pt* at the phone manufacturer and the camera manufacturer, respectively; see Listing 4.

The warehouse2 data node observes a case passing by; then, it generates a trace record for the case. In addition, it also observes that the phone and the camera are packed in the case. Therefore, it appends the packing information to the

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

```
Prefix(pt:=<http://cybermatics.org/owl/product−tracing/>)
...
ClassAssertion( pt:TraceRecord :warehouse2_record1 )
ObjectPropertyAssertion(pt :hasProduct :warehouse2_record1 :case)
DataPropertyAssertion( pt:beginTime :warehouse2_record1
    ''2016−04−11T00:00:00''^^xsd:dateTime)
DataPropertyAssertion( pt:endTime :warehouse2_record1
    ''2016−04−12T00:00:00''^^xsd:dateTime)
ClassAssertion( pt:PackRecord :packRecord )
ObjectPropertyAssertion(pt :hasPackageNode :packRecord :warehouse2)
ObjectPropertyAssertion( pt:hasCase :packRecord :case)
ObjectPropertyAssertion( pt:hasPacked :packRecord :phone)
ObjectPropertyAssertion(pt :hasPacked :packRecord :camera)
```

Fig. 6. The trace record of the case.

trace record. The ontological expression of this piece of information is shown in Fig. 6. The PackRecord class has a data property hasPackageNode, indicating the name of the data node where the packing information is collected. We assume that warehouse2 also adopts a uID system and issues a ucode for the record it collects. Since this ontology involves three products, warehouse2 registers the ucode of this record at the phone manufacturer, the camera manufacturer and the case manufacturer.

5.2. The ontology aggregation procedure

Finally, the retailer receives the phone, and it wants to determined how the phone arrived. The application terminal first scans the RFID tag attached to the phone to obtain the ID. Then, by the resolution service of the EPC network, the terminal obtains the address of the query node of the phone manufacturer and sends an aggregation request to the ontology URI query service. The request content includes the product's ID and the namespace for product tracing, i.e., "http://cybermatics.org/owl/product-tracing/ ".

Once the query node receives the request, it generates the URI list, which is composed of all the URIs that have been registered with the phone's ID and the specific namespace. Then, the query node responds to the terminal with the OnURI list.

The terminal receives the response. Then, it checks the OnURI list. If the OnURI list is not empty, it concurrently initiates requests for the information addressed by the URIs; otherwise, it stops collecting further information. In this case, the list is not empty, and the URIs include http URL and ucodes. Therefore, the network client uses the corresponding network protocols to request information. The requests are initiated concurrently, and thus, the network transmission time does not increase with the number of URIs.

When each data node receives a request, it responds with the information addressed by the URI to the terminal.

Thus far, the terminal has collected all the information that involves the phone. The information contains the product description, two trace records of the phone, a trace record of the case, and a packing record. Note that what the retailer employee wants is to trace the phone. Therefore, before we present the collected information to the employee, the terminal needs to exclude the product description and understand that the trace record of the case is also a trace record of the phone.

We define a class name TraceRecord_phone for the phone. By the SWRL rule defined below, Pellet infers that any trace record has a product with a URI of "urn:epc:1:2.24.400" is an instance of TraceRecord_phone.

```
TraceRecord(?t)^Product(?p)^hasProduct(?r,?p)^
    hasID(?p,"urn:epc:1:2.24.400"^^xsd:anyURI)−>TraceRecord_phone(?t)
```

However, the trace record of the case in Fig. 6 is also indispensable for tracing the phone. Then, another SWRL rule is defined as follows, which states that any trace record that has a case also has the products packed in the case if its node is the same as the package node. With this rule, Pellet can infer that the trace record that has the case in Fig. 6 has the phone. Combined with the above rule, Pellet can select all the trace records for the phone.

```
hasNode(?r, ?n) ^ hasCase(?rt, ?c) ^ hasPacked(?rt, ?p) ^ Node(?n) ^
hasPackageNode(?rt, ?n) ^ hasProduct(?r, ?c) −> hasProduct(?r, ?p)
```

If someone wants to trace the camera, they can define a similar class and a rule for the camera. The inference result is shown in Fig. 7, which indicates that all the trace records have been determined.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

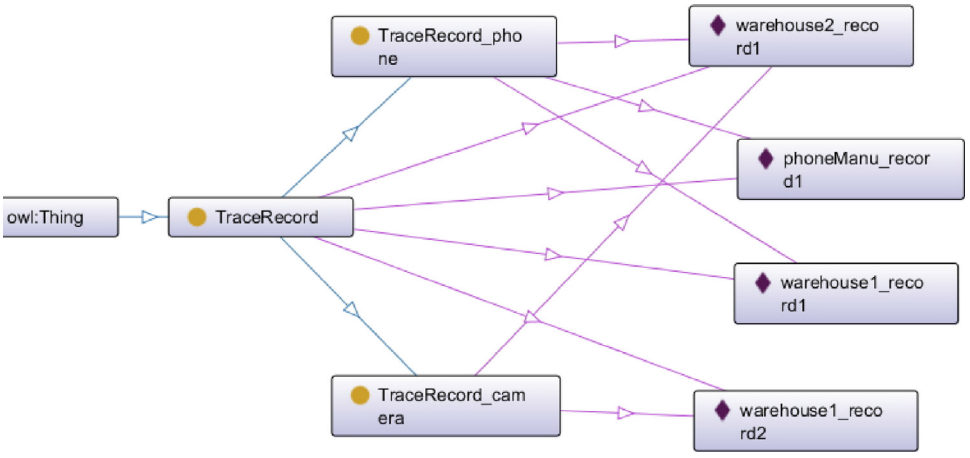

Fig. 7. The inferred individuals of TraceRecord drawn by protégé OntoGraf.

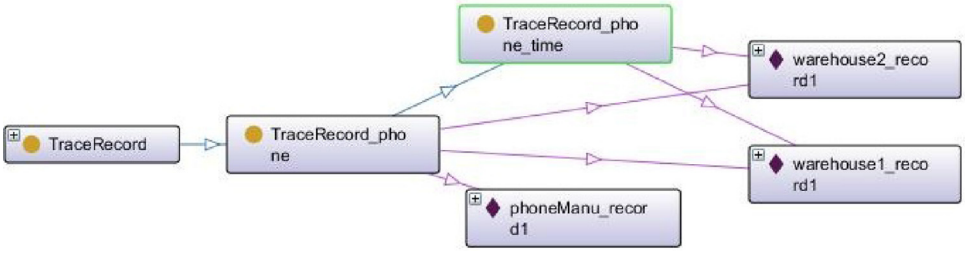

Fig. 8. The result of product tracing drawn by protégé OntoGraf.

However, according to the definition of production tracing, the records that do not satisfy the time constraint should be eliminated. Assuming that the retailer employee wants the record from Apr. 1, 2016, to Apr. 30, 2016, the constraint rule is as follows.

```
beginTime(?r, ?start) ^ endTime(?r, ?end) ^
    swrlb:greaterThanOrEqual(?end, ''2016−04−01T00:00:00''^^xsd:dateTime) ^
    TraceRecord_phone(?r) ^ swrlb:lessThanOrEqual(?start, ''2016−05−01T00:00:00''^^xsd:dateTime)
    −> TraceRecord_phone_time(?r)
```

Finally, from the result presented in Fig. 8, it can be observed that Pellet has successfully excluded phoneManu_record1, and the task of tracing the phone has been completed.

6. Conclusion and future directions

In conclusion, the architecture proposed in this paper is a potential solution to the task of aggregating information from multiple data nodes, which is common in many IIoT applications. The architecture is scalable, because each query node (manufacturer) only takes responsibility for its own products, and each data node only for the data collected by itself. The architecture is also efficient. Query/Register requests are responded instantly, because the addresses of related nodes can be directly resolved with the product URIs. Information and node heterogeneity is addressed in the architecture with semantic and service-oriented technologies. We also propose to use ontology namespaces as the condition to select the interesting ontologies. This scheme can reduces the network traffic and would be a flexible mechanism to share data among different applications.

In the case study section, taking the problem of product tracing as an example, we show how to apply the proposed architecture to applications in IIoT. In this case, we detail the ontology generation and register procedure and the ontology aggregation procedure. The focus of this section is the use of semantic technologies, and it is shown that the reasoning ability of semantic technologies has the potential to support various important functionalities and thus to lay a solid foundation to support heterogeneous IIoT applications.

There are several important directions for future works. We have pointed out some disadvantages of the proposed architecture. The data availability is not solid, and in some cases, the network traffic could be heavy. Strategies are needed to

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

relieve these problems. Moreover, before placing the architecture into practice, the security and privacy problems should be taken into consideration. In addition, evaluating the proposed architecture in other applications is also interesting.

Acknowledgements

This work is supported by the Fundamental Research Funds for the Central Universities (No. 06116073, No. 06105031) and the National Natural Science Foundation of China (No. 61471035).

References

[1] Miorandi D, Sicari S, De Pellegrini F, Chlamtac I. Internet of things: Vision, applications and research challenges. Ad Hoc Netw. 2012;10(7):1497–516. doi:10.1016/j.adhoc.2012.02.016.
[2] Atzori L, Iera A, Morabito G. The internet of things: A survey. Comput. Netw. 2010;54(15):2787–805. doi:10.1016/j.comnet.2010.05.010.
[3] Riazul Islam SM, Daehan Kwak, Humaun Kabir M, Hossain M, Kyung-Sup Kwak. The internet of things for health care: A comprehensive survey. IEEE Access 2015;3:678–708. doi:10.1109/ACCESS.2015.2437951.
[4] Santos DFS, Almeida HO, Perkusich A. A Personal connected health system for the internet of things based on the constrained application protocol. Comput. Electr. Eng. 2014;44:122–36. doi:10.1016/j.compeleceng.2015.02.020.
[5] Fang S, Xu L, Zhu Y, Liu Y, Liu Z, Pei H, et al. An integrated information system for snowmelt flood early-warning based on internet of things. Inf. Syst. Front. 2015;17(2):321–35. doi:10.1007/s10796-013-9466-1.
[6] Agrawal R, Cheung A, Kailing K, Schönauer S. Towards traceability across sovereign, distributed RFID databases. Proc. Int. Database Eng. Appl. Symposium, IDEAS 2006:174–84. doi:10.1109/IDEAS.2006.47.
[7] Qin E., Long Y., Zhang C., Huang L.. Cloud computing and the internet of things: Technology innovation in automobile service. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)2013; 8017 LNCS (PART 2):173–180.
[8] Xu LD, He W, Li S. Internet of things in industries: A survey. IEEE Trans. Ind. Inf. 2014;10(4):2233–43. doi:10.1109/TII.2014.2300753.
[9] Pang Z, Chen Q, Han W, Zheng L. Value-centric design of the internet-of-things solution for food supply chain: Value creation, sensor portfolio and information fusion. Inf. Syst. Front. 2012;17(2):289–319. doi:10.1007/s10796-012-9374-9.
[10] Chen RY. Autonomous tracing system for backward design in food supply chain. Food Control 2015;51:70–84. doi:10.1016/j.foodcont.2014.11.004.
[11] Koshizuka N, Sakamura K. Standards & emerging technologies ubiquitous ID. Context 2010;9(4):98–101. doi:10.1109/MPRV.2010.87.
[12] Wu Y, Ranasinghe DC, Sheng QZ, Zeadally S, Yu J. RFID Enabled traceability networks: A survey. Distrib. Parallel Databases 2011;29(5–6):397–443. doi:10.1007/s10619-011-7084-9.
[13] Wu Y, Sheng QZ, Shen H, Zeadally S. Modeling object flows from distributed and federated RFID data streams for efficient tracking and tracing. IEEE Trans. Parallel Distrib. Syst. 2013;24(10):2036–45. doi:10.1109/TPDS.2013.99.
[14] Hitzler P, Körtzsch M, Rudolph S. Foundations of semantic web technologies. Chapman & Hall/CRC; 2009.
[15] Barnaghi P, Wang W, Henson C, Taylor K. Semantics for the internet of things: Early progress and back to the future. Int. J. Semantic Web Inf. Syst. 2012;8:1–21. doi:10.4018/jswis.2012010101.
[16] Chen L, Nugent C, Mulvenna M, Finlay D, Hong X. Semantic smart homes: Towards knowledge rich assisted living environments. Stud. Comput. Intell. 2009;189:279–96. doi:10.1007/978-3-642-00179-6_17.
[17] Skillen KL, Chen L, Nugent CD, Donnelly MP, Burns W, Solheim I. Ontological user modelling and semantic rule-based reasoning for personalisation of help-On-Demand services in pervasive environments. Future Gener. Comput. Syst. 2014;34:97–109. doi:10.1016/j.future.2013.10.027.
[18] Sheth AP, Larson JA. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. 1990;22(3):183–236. doi:10.1145/96602.96604.
[19] Cheung A, Railing K, Schönauer S. Theseos: A query engine for traceability across sovereign, distributed RFID databases. Proc. Int. Conf. Data Eng. 2007:1495–6. doi:10.1109/ICDE.2007.369050.
[20] Dressler F. A Study of self-organization mechanisms in ad hoc and sensor networks. Comput. Commun. 2008;31(13):3018–29. doi:10.1016/j.comcom.2008.02.001.
[21] Guinard D, Trifa V, Karnouskos S, Spiess P, Savio D. Interacting with the SOA-based internet of things: discovery, query, selection, and on-demand provisioning of web services. IEEE Trans. Serv. Comput. 2010;3(3):223–35. doi:10.1109/TSC.2010.3.
[22] Lu J, Ma L, Zhang L, Brunner J-s, Wang C, Pan Y, et al. SOR : A practical system for ontology storage , reasoning and search. In: Proceedings of the 33rd international conference on Very large data bases; 2007. p. 1402–5. ISBN 9781595936493.
[23] Oren E, Kotoulas S, Anadiotis G, Siebes R, ten Teije A, van Harmelen F. Marvin: Distributed reasoning over large-scale semantic web data. J. Web Semantics 2009;7(4):305–16. doi:10.1016/j.websem.2009.09.002.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018

Tao Zhu received his Ph.D. degree from the University of Science and Technology of China in 2015 and his BE degree from Central South University in 2009. Currently, he works as a post-PhD and lecturer in the School of Computer & Communication Engineering, University of Science & Technology Beijing. His research interests include Cybermatics, Internet of Things and Evolutionary Computation.

Sahraoui Dhelim received his Bachelor degree in fundamental computer science from the University of Djelfa in 2012 and his Master degree in networking and distributed systems from the University of Laghouat in 2014. He has been pursuing a Ph.D. at the University of Science and Technology Beijing since 2015.

Zhihao Zhou received his B.E. degree from Shandong Technology and Business University, China, in 2015. He is currently pursuing a M.S. degree at the University of Science & Technology Beijing, China. His current research interests include Internet of Things, object modelling in smart homes, and the semantic web.

Shunkun Yang, associate research professor at Beihang University, received his B.S., M.S., and Ph.D. degrees from the School of Reliability and Systems Engineering from Beihang University in 2000, 2003, and 2011, respectively. He also worked as an associate research scientist at Columbia University from 2014.09 to 2015.09. His current research interests include reliability, testing and diagnosis, and so forth.

Huansheng Ning received his Ph.D. degree from Beihang University in 2001. He is currently a professor in the School of Computer & Communication Engineering, University of Science & Technology Beijing. He previously was a post-PhD and associate professor in the School of Electronic and Information Engineering, Beihang University. He is a Co-Chair of the IEEE SMC Society TC on Cybermatics.

Please cite this article as: T. Zhu et al., An architecture for aggregating information from distributed data nodes for industrial internet of things, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.018