Sensing as a Service- A Service-Oriented Collaborative Sensing Framework for Detecting Composite Events in Industrial Cyber-Physical System

Tao Wang[1], Hong Xiao[2] and Lianglun Cheng[2]
[1] *Faculty of Automation, Guangdong University of technology, Guangzhou, China*
[2] *Faculty of Computer, Guangdong University of technology, Guangzhou, China*

Keywords: Sensing as a service, RESTful sensing service, Collaborative sensing, Composite event, Service composition, Industrial cyber-physical system.

Abstract: The widespread deployment of sensors in Industrial Cyber-Physical System (ICPS) enables real-time monitoring kinds of composite events occurring in the industrial process. However, due to the high degree of heterogeneity of the sensing nodes in ICPSs, it is very hard to effectively composing the platform-specific functionalities provided by heterogeneous sensor nodes for detecting various composite events. In this article, by exploiting REST framework and expanding the basic architecture of S2aaS, we propose a service-oriented and lightweight collaborative sensing framework for detecting multiple composite events in ICPSs. Further, the specific implementation details about the process of RESTful service registry of sensing node, sensing service discovery and service composition is presented. We develop an application prototype to test the feasibility and scalability of the system, and the experiment results show that RESTful-based sensing service collaboration outperforms SOAP-based one with more lightweight communication and less power consumption as expected. Based on our proposed collaborative sensing framework with Restful sensing services, it is very convenient to provide a sensor web services for various requirements of composite event detection.

1 INTRODUCTION

Recently, the widespread deployment of RFIDs, sensors, wireless sensor networks, and embedded systems has fostered the rise of industrial Cyber-Physical Systems (ICPS), which is considered as transformative technologies for managing interconnected systems between industrial physical assets and computational capabilities. ICPS is the basic premise for the implementation of industry 4.0. By integrating the emerging information and network technologies (e.g., data sensing, network transmission, high-performance computing, big data, intelligent decision-making and controlling), ICPS is able to greatly improves the performance on real-time interaction, efficient collaboration and dynamic optimization for the industrial system, and create a new industrial manufacturing and information service mode.

With the sensing devices deployed in industrial lines, it is able to monitor the real-time status of the industrial process for high quality. Specially, we are very concerned about the composite events occurring in the industrial process, e.g., abnormal changes in the industrial production environment, abnormal working status of mechanical equipment, and the detection of such composite events is very helpful for realizing dynamic feedback controlling and scheduling on the industrial process. Unlike an atomic event which depends on single-mode sensing data, a composite event is a combination of several atomic events and its occurrence is jointly determined by collaborative sensing with heterogeneous sensors (Chen, 2015). For example, in order to identify an abnormal working status of mechanical equipment, maybe it is necessary to collect the following multi-modal sensing data from video sensors, temperature/moisture sensors, displacement sensors, vibration sensors, etc.

In general, with the widespread deployment of RFIDs/ RFID readers and sensors in ICPSs, it is able to detect kinds of composite events for obtaining more meaningful information within a factory. On the

Wang, T., Xiao, H. and Cheng, L.
Sensing as a Service- A Service-Oriented Collaborative Sensing Framework for Detecting Composite Events in Industrial Cyber-Physical System.
DOI: 10.5220/0008099402640271
In *Proceedings of the International Conference on Advances in Computer Technology, Information Science and Communications (CTISC 2019)*, pages 264-271
ISBN: 978-989-758-357-5

one hand, each composite event is determined with the data from a set of heterogeneous sensing devices, in the other hand some sensing devices could be exploited simultaneously for detecting multiple composite events. Various different types of composite events are the consumers of sensor data, so we need easy and feasible mechanisms to access the large-scale distributed sensor devices in ICPSs. However, in such environments, due to the device heterogeneity and differing accompanying protocols, integrating diverse sensing devices into observation systems for detecting multiple composite events is not straightforward (Guinard, 2010). Therefore, it is crucial to build a coherent infrastructure which treats sensors in an interoperable, platform-independent way.

In recent years, several efforts have been invested in order to handle the challenges related to the integration of large-scale heterogeneous sensing devices in an interoperable and uniform way. Sensing as a service (S^2aaS) is introduced to provide sensing services using kinds of sensors via a cloud computing system (Sheng, 2013). In the S^2aaS cloud, heterogeneous sensing devices are abstracted as services and expose their functionalities with common accessing interfaces and encodings. Web Services (Hoang, 2012) are proposed to provide a standard and interoperable accessing means for heterogeneous sensing devices. Because of the constrained resources in the context of the ICPS, the RESTful Web Services (Garriga, 2016) have been exploited in some studies for many advantages over arbitrary Web Services (i.e., SOAP), such as less overhead, less parsing complexity, statelessness, and tighter integration with HTTP. Therefore, with the architecture of S^2aaS and RESTful Web Services, it is able to provide high-level abstraction for the development of detecting multiple composite events in ICPSs.

There are some related researches on building lightweight REST framework for wireless sensor networks. In the study of environmental sensor monitoring (Lee, 2014), Restful Web Service is used for communication with the Arduino-based sensors. To address the constrained resources in sensor nodes, Rouached et al. (Rouached, 2012) propose a lightweight RESTful approach based on Restful SWE services for interacting with the constrained WSNs. Taherkordi et al. (Taherkordi, 2011) apply REST concepts to develop Web services for WSNs and smartphones as two representative resource-constrained platform. The above studies have provided a framework for developing Restful web services for sensor nodes, however, there are also few researches on how to efficiently combine kinds of Restful sensing services for detecting various composite events.

In this paper, by exploiting REST framework and expanding the basic architecture of S^2aaS, we study to present a service-oriented and lightweight collaborative sensing framework with more specific implementation details for detecting multiple composite events in ICPSs. Each sensing device register its service in the sensing cloud platform. When a user initiates a request of detecting a composite event, the proposed framework will automatically resolve the request into a set of sensing tasks and then push it to a subset of sensing devices that happen to be in the area of interest. Based on our proposed collaborative sensing framework with Restful sensing services, it is very convenient to provide a sensor web services for various requirements of composite event detection, enable users to connect and share the heterogeneous sensor resources more efficiently.

The rest of this paper is organized as follows. The Restful sensing services and composite event model are described in Section 2. In Section 3, we present the service-oriented collaborative sensing framework. In Section 4, an automatic sensing service composition process for detecting a composite events is introduced. We provide the experimental results in Section 5 and conclude the paper in Section 6.

2 RESTFUL SENSING SERVICES

2.1 Restful Web Service for Sensing Nodes

Representational State Transfer (REST) is an architectural model for building distributed applications flexibly (Pautasso, 2008). It exploits the natural structure of the Web and is efficiently implemented with the Hypertext Transfer Protocol (HTTP). Considering the constrained resources of sensing nodes in ICPS, RESTful web service is suitable to be used in ICPS due to its lightweightness and its resource-oriented conception. In RESTful architecture, every sensing sources can be uniquely identified as a URI, for example, a URI for a vibration sensor on a sensing node which monitors the status of the equipment *m* located in workshop *n* is shown as follows:

/{Location}/sensingnode/{n_id}/sensor/vibration, here, location= /workshop_n/equipment_m.

Based on the unique identification through URIs, a node is able to provide a RESTful Web service with

a uniform interface by using standard HTTP methods (GET, PUT, POST, DELETE).The sensing node is embedded with a lightweight web server with basic HTTP functionality, e.g., NanoHTTPD (Elonen, 2018), a tiny web server which has been proven to be suitable for the resource constrained environment. We can obtain the data from the light sensor by the following HTTP GET request:

GET /{Location}/sensingnode/{n_id}/sensor/light HTTP/ 1.1
Content-type: application/json

Although the whole set of standard HTTP methods could be used, there are some resources which don't offer all of them. Some HTTP methods is not able to be matched to any functionality of the corresponding resources, e.g., no functionality of the temperature sensor that could be matched on a DELETE action.

As shown in Figure 1, for a sensing node deployed with several sensors, we present a hierarchical organization of the sensing resources, which are in turn helpful for generating the URIs of sensing resources. We can notice that a sensing node may generally contain the following resources: some sensors for provide sensing data, LEDs that offers pulling and changing its state, battery, and tasks that can be created, supervised, altered and deleted. Unlike sensors and LEDs which are physical entities, a task is a logical resource. Given the parameters of execution frequency, threshold and target sensor, a task is able to periodically check the corresponding sensor value against a certain threshold and then alert if the threshold is exceeded.

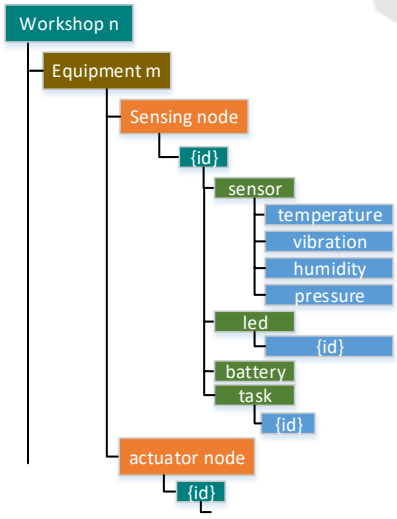

Figure 1: Hierarchical organization of sensing resources.

Therefore, each service provide by a sensing node can be addressed and accessed with a unique URI. URIs provide a global naming scheme which allows lightweight service discovery. With the URIs, kinds of resources can be organized more effectively and helpful for building an automatic service discovery and composition framework.

2.2 The Composite Event Model

Events are classified into atomic events and composite events. An atomic event denotes an observable occurrence of a phenomenon or an object reflected by a single sensing value, e.g., the temperature of the workshop exceeds a warning threshold. An atomic event can be represented by $e(t, l, R)$, where t is the occurrence time of the event and it could be a time-interval or a time point, l is the occurrence location which could be a specific point or field in the workshop, R is expressed by a logic expression which denotes the condition of the event occurring. For example, e(*14/06/2017*, */workshop_n/equipment_m, temperature>60℃*) expresses that the temperature of the equipment m on *14/06/2017* is greater than 60℃.

A composite event characterizes an observable occurrence of a complex phenomenon or an object, and it is composed by several atomic events with the specific temporal and spatial constraints. Given an atomic event $e_i(t_i, l_i, R_i)$, a model of the composite event can be described as follows (Gao, 2015).

$$E(R_1 \wedge R_2 \wedge ... \wedge R_k \wedge C_t \wedge C_l, \delta) = E((e_1, \delta_1), (e_2, \delta_2), ..., (e_k, \delta_k), C_t, C_l, \delta) \quad (1)$$

where δ_i $(0 \leq \delta_i \leq 1)$ is the confidence of an atomic event e_i which indicates the occurrence probability of the composite event E when e_i is occurring. C_t and C_l respectively represent the temporal and location constraints on the atomic events. R_i defines the occurrence conditions of the atomic events. The parameter δ could be computed from $\{\delta_1, \delta_2, ..., \delta_k\}$, and it denotes the occurrence confidence of the composite event E when all the related atomic events happen. Therefore, based on the formula (1), each composite event can be parsed to be several atomic events with the same temporal and location constraints and different event occurring conditions.

3 SERVICE-ORIENTED COLLABORATIVE SENSING FRAMEWORK

In this section, in order to provide a flexible platform for building the applications for detecting composite events, we introduce a service-oriented collaborative sensing framework based on the RESTful web services of sensing nodes. The framework shown in Figure 2 consists of several subsystems and modules. The first subsystem Rest-Based Sensing Node is composed by kinds of heterogeneous sensing devices with different operation systems (e.g., Z-Stack, Tiny OS, Contiki) and communication methods (e.g., WiFi, IEEE 802.15.4). These sensing devices are equipped with a tiny web server supporting the basic web service interface, and then provide lightweight RESTful web services with HTTP methods. Each sensing node registers its services with the following root URI: */{Location}/sensingnode/{n_id}*.

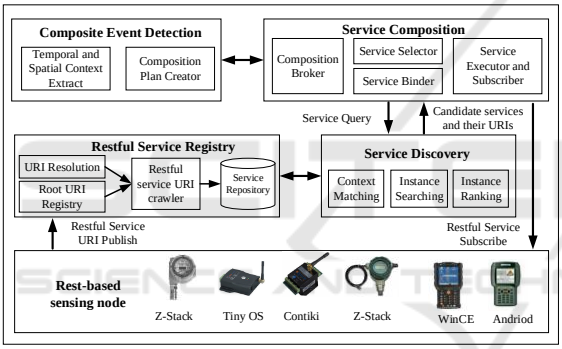

Figure 2: A service-oriented collaborative sensing framework based on the RESTful web services.

Service Registry is the second subsystem which is responsible for handling the register requests of each node, and then gets all the service URIs with a web service crawler. The URI Resolution module in this subsystem resolves the URIs of RESTful web services and then obtains its functionality and spatial property, which are helpful for the implementation of service discovery. The third subsystem Service Discovery has three main modules including Context Matching, Instance Search and Instance Ranking, and finds out candidate services and their URIs from Service Cache according to kinds of service queries from the subsystem of Service Composition.

The fourth and central subsystem Service Composition is mainly responsible for providing composition services in the framework. The Composition Broker receives a request with composition plan of a composite event detection tasks, and then deliver an appropriate composition service by calling three functional modules including Service Selector, Service Binder, Service Executor and Subscriber. The fifth subsystem Composition Event Resolution is also an important part, which is playing as a link between the service composition subsystem and the user who launches the detection request of composite event. This subsystem is functionally divided into two main modules including Composition Plan Creator, Temporal and Spatial Context Extractor.

3.1 Service Registry

All the Rest-based sensing nodes publish the services with their root URIs, e.g., */{Location}/sensingnode/{id}*. With the root URIs of sensing nodes, the Restful service URI crawler sends a request for obtain the specific service description of each sensing node with the hRESTS microformat. Then, by using MicroWSMO, which is a lightweight semantic service description approach based on WSMO-Lite service ontology, the service resolution module is able to add semantic information on top of hRESTS service document. With the WSMO-Lite service ontology, a sensing service can be described with the following four types of service semantics: information semantics, functional semantics, non-functional semantics and behavioral semantics.

We can then extract RDF data from the MicroWSMO semantic descriptions for the RESTful services. The RDF data could be mapped with the four types of semantics (functional, nonfunctional, behavioral and information) defined by WSMO-Lite, and then a RESTful service can be described with RDF data as follows.

RESTful Service

Functional semantics:	
Has_ServiceId:	*onto:restws#ServiceId*
Has_ServiceLabel:	*onto:restws#Label*
Has_Provider:	*onto:restws#NodeId*
Nonfunctional semantics:	
Has_ServiceLocation:	*onto:restws#ServLocation*
Has_ServiceTime:	*onto:restws#ServTime*
Has_BatteryLevel:	*onto:restws#BatteryLevel*
Has_cost:	*onto:restws#Cost*
Behavioral semantics:	
Has_OperationId:	*onto:restws#OpId*
Has_OperationLabel:	*onto:restws#OpLabel*
Has_OpMethod:	*onto:restws#OpMethod*
Has_URI:	*onto:restws#URI*
Information semantics:	
Has_input:	*onto:restws#OpInput*
Has_output:	*onto:restws#OpOutput*

3.2 Service Discovery

Given a service query $S_Q=\{F_Q,N_Q,I_Q\}$ from the service composition module, the service discovery module returns an appropriate service instance $S_I=\{F_I,N_I,B_I,I_I\}$ by matching the query with the huge amount of candidate service instances. As shown in Figure 3, we present a multi-stage semantic service discovery process, which includes the following steps: 1) service type lookup based on functional and information semantics; 2) service filtering by context matching with nonfunctional semantics (mainly with spatial and temporal contexts); 3) service ranking with some key factors from the service nonfunctional semantics (e.g., cost, residual battery level), and service monitor module collects the dynamic nonfunctional information.

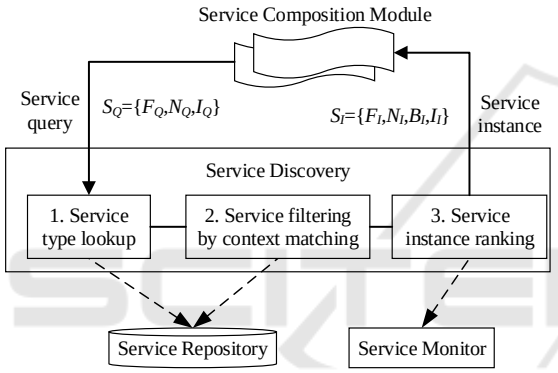

Figure 3: A multi-stage service discovery process.

3.2.1 Service Type Lookup

This the first stage of service discovery process. The service type lookup module matches the service query and the service instances based on the functional and information semantics, and it selects out a set of appropriate RESTful services with the same service type of the service query. However, because the process of matching semantic ontology concepts among all the individual service instances has typically been time and resource intensive, some efficient service organization model and matching algorithms should be adopted.

As shown in Figure 4, a service organization and Matching with Dynamic Bloom Filter (DBF) is exploited. Firstly, the RESTful semantic services in the service cache is categorized with different service types, and each service type is mapped into a bloom filter with m bits which manages kinds of service types. With a number of j hash functions $h_j(\ )$, the functional and information semantics of a service instance are hashed into random numbers between 1 and m, and plus one to the numerical numbers of corresponding bits. Then, the service instances which have the same hash operation results are clustered together.

Secondly, each cluster of service instances with the same service type is managed with a dynamic bloom filter including multiple static bloom filters with m bits. The service instances which have the same contexts (mainly refer to spatial and temporal contexts) are managed with a static bloom filter.

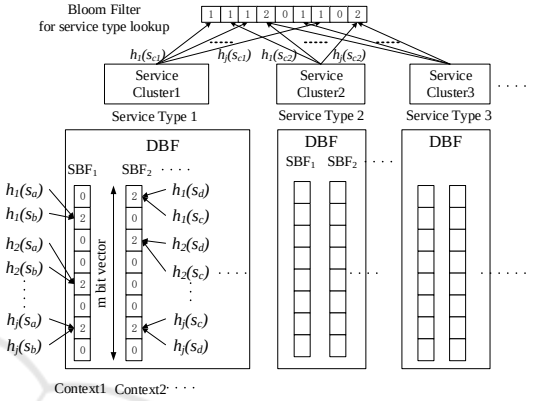

Figure 4: Service organization and Matching with Dynamic Bloom Filter.

3.2.2 Service Filtering by Context Matching

This module is helpful for reducing the service exploration space greatly, and it is a lightweight service filtering process based on a small set of contexts (mainly on spatial and temporal contexts, e.g., service location, service time) which are extracted from the service nonfunctional semantics. In order to improve the efficiency and success ratio of service discovery, the context matching module filters out the RESTful semantic services of which the spatial and temporal contexts do not match the contexts of the service query apparently. For example, in order to monitor a dangerous gas leakage which may happen at workshop *A*, the services provided by the sensors far away from workshop *A* should be filtered out first.

Besides, depending on their value types, the contexts are generally classified into numeric and instance contexts. For example, the service location "workshop A" is an instance and spatial context, and the service time "*14/06/2017*" is a numeric and temporal context. As shown in Figure 5, the service instances with the same spatial and temporal contexts are managed by the same static bloom filter. Then, the service location and service time of a service query and service instances are matched by the static bloom filters contained by a DBF with the same service type.

After service type lookup and service filtering by context matching, we can obtain a set of service instances that match the functionality and context requirements of a service query.

3.2.3 Instance Ranking

In order to get a most appropriate service instance from the candidate service instances selected out by the above steps, this module ranks the candidate service instances based on their service cost, quality of service, residual battery level, etc. The output of the ranking process is an ordered list of service instances corresponding to the non-functional requirements expressed by the service query. Generally, those service instances that have high battery level and low service cost get higher rankings.

4 AUTOMATIC SENSING SERVICE COMPOSITION FOR DETECTING A COMPOSITE EVENTS

4.1 Composition Plan Description Language (CPDL) for Composite Event

For a composite event detection request, we need make a composition plan for it. As described in section 3.2, a composite event is denoted as $E(R_1 \wedge R_2 \wedge ... \wedge R_k \wedge C_t \wedge C_l, \delta)$, and R_i denotes the condition of an event occurring which is monitored by a sensing service S_i. For example, an atomic event *temperature>60℃* should be monitored by a temperature sensor with its service *SensingTemperature*. In order to distribute a composite event detection request to kinds of sensing services, here, the language called Composition Plan Description Language (CPDL) (Han, 2014) has been designed to describe composition plans.

An example of a CPDL document for a fire event detection is shown in Listing 1. This document describes a composition plan with the context requirements including a specific location and a specific data acquisition frequency. It also describes the composite service consisting of three component services *SensingTemperature*, *SensingLight* and *SensingHumidity* which detect the related atomic events.

```
1.  <CSDL xmlns:xsi=http://xxxx/SaaS>
2.    <context location=workshop_n, data_freq=5 seconds>
3.      <service> SensingTemperature</service>
4.      <service>SensingLight</service>
5.      <service>SensingHumidity</service>
6.  </context>
7.  </CSDL>
```

Listing 1. Composition Plan Description Language (CPDL) for a fire event detection

4.2 Automatic Sensing Service Composition

The service composition plan of a composite event detection is then transferred to the service broker module in the Service Composition subsystem. The service broker invokes three functional modules to accomplish the task of service composition, and the specific process is described as follows:

1） Service Query: The service broker resolves the CPDL for a composite event detection request, and extracts the specific contexts and component services defined by the CPDL. For example, the location *workshop_n* and the required component service *SensingLight* can be extracted. Then, the service queries are sent to the service discovery subsystem and then waiting for the returned candidate service instances and their URIs.

2） Service Selection: With the candidate service instances for each required component service, this module selects the appropriate combination according to the overall non-functional requirements of the composite event detection, such as available battery level, service cost. Therefore, there will be a combinatorial optimization problem for selecting service instances for detecting multiple atomic events, and the intelligent particle swarm optimization algorithm is exploited to handle this problem.

3） Service Binding: This module binds the selected service instances to their operations. For example, the service *SensingLight* provided by a sensor node 1 in workshop 1 is bound to the following operation:
GET http://*workshop_1*/sensingnode/{1}/sensor/light

4） Service Executor and Subscriber: This module is responsible for executing the whole process of service composition, and sends the requests according to the defined operations. Generally, there are two ways to get sensor data: the request/response mode is always used to get sensor data once, and the subscription/

publishing mode is always suitable for gathering data periodically.

5 PROTOTYPE AND EXPERIMENTS

As shown in Figure 5, a system prototype was developed to illustrate the operation of the service-oriented collaborative sensing system and to test the feasibility and scalability of the system. There are kinds of sensor nodes which are deployed in two workshops, and the sensor nodes includes temperature and humidity sensor node, light sensor node, smoke sensor node, CO2 sensor node, CCD node, and so on. The sink node is responsible for forwarding the sensing data from the sensor nodes to the platform and the operation commands from the platform to the sensor nodes. All the sensor nodes are equipped with a lightweight web server NanoHTTPD [11], and respond kinds of HTTP requests in JSON format. The sensor nodes register their ID and service capabilities at the platform. The gateway is mainly responsible for data forwarding and network protocol conversion. The platform of SaaS (Sensing as a service) in Figure 5 implements the subsystems defined in Figure 3, including service registry, service discovery and service composition. The platform receives various requests of composite event detection from web clients and returns the related messages when the composite events happen and have been detected.

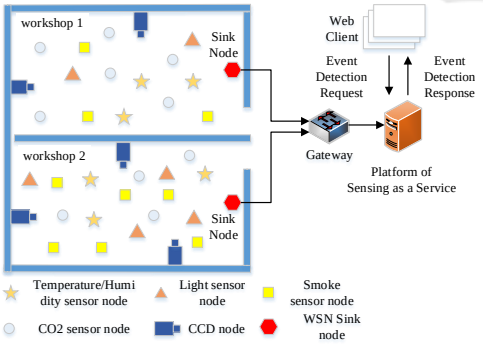

Figure 5: A system prototype for verifying the proposed service-oriented collaborative sensing framework.

In order to evaluate the feasibility and scalability of the proposed system, the following experiments were carried out to verify the service discovery, service selection as well as service composition and execution processes. Firstly, we measure the cost of data transmission for the proposed Restful service composition method. For the request/response process of the same component service (e.g., *SensingLight*), we study the performance on data transmission duration by comparing different technologies of web services including REST/JSON, SOAP/XML and REST/XML, and the experiment results are shown in Figure 6. It is clear that the reduction of the transmission time is more obvious by using REST instead of SOAP. Even for REST, with JSON format the results are better than with XML since XML is verbose.

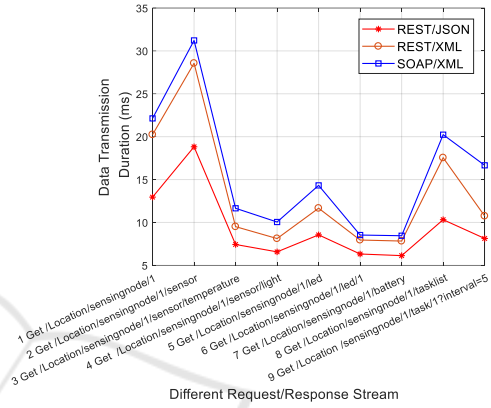

Figure 6: Data transmission duration – REST/JSON vs REST/XML vs SOAP/XML.

As shown in Figure 7, we also evaluate the power consumption of getting list of all services on a specific sensing node when using different technologies of web services including REST/JSON with SOAP/XML. As it is expected, SOAP/XML-based Web service consumes much more energy, especially for data communication. The CPU power consumption due to parsing and processing SOAP messages is ignorable when compared with the power usage of radio.

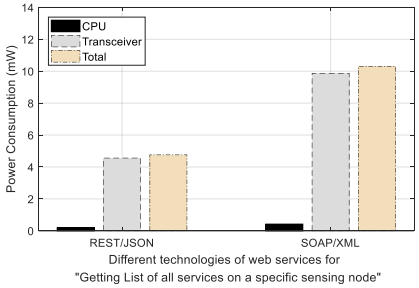

Figure 7: Power consumption for different technologies of web services.

The service discovery and selection process is the most time-consuming part in the whole service

composition process. Therefore, the second set of experiments aims to measure the time performance of service discovery and selection as the number of sensor nodes increases from 50 to 500, and the results are shown in Figure 8. We can find that the time of service composition keeps under 150ms even in a critical situation with the participation of 500 devices, and the stability of the system has been well guaranteed with the proposed service discovery and selection method.

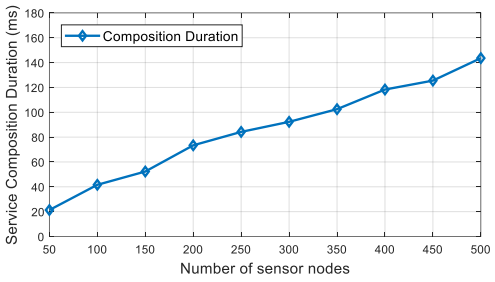

Figure 8: Duration of service composition process vs the number of sensor nodes increase from 50 to 500.

6 CONCLUSION AND FUTURE WORK

In this paper, by exploiting the technology of RESTful web service, we have proposed a loosely-coupled, lightweight and service-oriented collaborative sensing framework for detecting multiple composite events in ICPSs, and provides more specific implementation details about RESTful service registry, service discovery, service composition. We also present an application prototype and evaluate the performance of REST/JSON based web service by comparing with other technologies of web services including SOAP/XML and REST/XML. The experiment results show that RESTful-based sensing service collaboration outperforms SOAP-based one with more lightweight communication and less power consumption as expected. The future work will focus on how to optimize the process of service composition when handling concurrent requests of composite event detection.

ACKNOWLEDGEMENT

Our work is supported by Guangdong Provincial Key Laboratory of Cyber-Physical System as well as multiple funds in china, including the National Natural Science Foundation of China (61502110, 61672170), the Key Program of NSFC-Guangdong Joint Funds (U1801263, U1701262), Major projects of science and technology plan of Guangdong Province (2015B090922013, 2017A010101017, 2017B090901019,2016B090918045,2016B090918017).

REFERENCES

C. L. Chen, J. Yan, N. Lu, et al., 2015. Ubiquitous Monitoring for Industrial Cyber-Physical Systems Over Relay Assisted Wireless Sensor Networks. *IEEE Transactions on emerging topics in computing*, vol 3, no. 3, pp. 352-362.

D. Guinard, Vlad Trifa, et al., 2010. Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. *IEEE Transactions on services computing*, vol. 3, no. 3, pp. 223-235.

X. Sheng, J. Tang, X. J. Xiao, et al., 2013. Sensing as a Service: Challenges, Solutions and Future Directions. *IEEE Sensors Journal*, Vol. 13, No. 10, pp. 3733- 3741.

D. D. Hoang, H. Y. Paik, 2012. Service-Oriented Middleware Architectures for Cyber-Physical Systems. *IJCSNS International Journal of Computer Science and Network Security*, Vol.12 No.1, pp.79-87.

M. Garriga, C. Mateos, Andres Flores, et al., 2016. RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications*, 60, pp. 32-53.

S. C. Lee, J. Y. Jo, et al., 2014. Environmental Sensor Monitoring With Secure Restful Web Service. *International Journal of Services Computing*, Vol. 2, No. 3.

M. Rouached, S. Baccar, et al., 2012. RESTful Sensor Web Enablement Services for Wireless Sensor Networks, *IEEE Eighth World Congress on Services*, pp. 65-72.

A. Taherkordi, F. Eliassen, et al., 2011. Chapter 9: RESTful Service Development for Resource-Constrained Environments. *REST: From Research to Practice.*

C. Pautasso, O. Zimmermann, et al., 2008. Restful web services vs. "big" web services: making the right architectural decision. *In Proceedings of the 17th international conference on World Wide Web*, pp. 805-814, NY, USA.

J. Elonen. Nanohttpd. https://github.com/NanoHttpd/nanohttpd. (last accessed 24 Nov 2018)

J. Gao, J. Z. Li, Z. P. Cai, et al., 2015. Composite Event Coverage in Wireless Sensor Networks with Heterogeneous Sensors. *INFOCOM*, pp.217-225.

S. N. Han, G. M. Lee and N. Crespi, 2014. Semantic Context-aware Service Composition for Building Automation System. *IEEE Transactions on industrial informatics*, vol.10, no.1, pp.752-761.