A WS-Agreement Based SLA Ontology for IoT Services

Fan Li(✉), Christian Cabrera(✉), and Siobhán Clarke(✉)

Trinity College Dublin, College Green, Dublin, Ireland
{fali,cabrerac,Siobhan.Clarke}@scss.tcd.ie

Abstract. In the Internet of Things (IoT), billions of physical devices, distributed over a large geographic area, provide a near real-time state of the world. These devices' capabilities can be abstracted as IoT services and delivered to users in a demand-driven way. In such a dynamic large-scale environment, a service provider who supports a service level agreement (SLA) can have a comprehensive competitive edge in terms of service quality management, service customization, optimized resource allocation, and trustworthiness. However, there is no consistent way of drafting an SLA with respect to describing heterogeneous IoT services, which obstructs automatic service selection, SLA negotiation, and SLA monitoring. In this paper, we propose an ontology, WIoT-SLA, to achieve semantic interoperability. We combine IoT service properties with two prominent web service SLA specifications: WS-Agreement and WSLA, to take advantage of their complementary features. This ontology is used to formalize the SLAs and SLA negotiation offers, which further facilitates the service selection and automatic SLA negotiation. It can also be used by a monitoring engine to detect SLA violations by providing the semantics of service level objectives (SLOs) and quality metrics. To evaluate our work, a prototype is implemented to demonstrate its feasibility and efficiency.

Keywords: Internet of Things · SLA · Service level management · SLA ontology

1 Introduction

The Internet of Things (IoT) is an ambient smart environment where a large number of interconnected physical objects interact with the physical world, providing a near real-time state of the environment. Each device's functionalities can be abstracted as an IoT service provided through a well-defined interface in a homogeneous way [22]. For mission-critical IoT applications, "best effort" services are not sufficient [21]. In many service provisioning needs, SLAs are widely used as a contract to provide a certain level of control to a consumer

Supported by Science Foundation Ireland (SFI) under the project SURF - grant 13/IA/1885.

V. Issarny et al. (Eds.): ICIOT 2019, LNCS 11519, pp. 58–72, 2019.
https://doi.org/10.1007/978-3-030-23357-0_5

and deliver requested services with pre-negotiated quality of service (QoS). In SLAs, the obligations and guarantees of involved parties are specified in the form of Service Level Objectives (SLOs), which are evaluated using measurable data [16].

To our best knowledge, SLA management in IoT platforms is still in a preliminary stage [17]. However, providing a precise SLA specification for IoT services would enable a better QoS-aware service management [12]. For example, the varying syntax of different SLAs obstructs automatic service matching and SLA negotiation in large-scale electronic markets. Consumers or third-party audit agents struggle to detect SLA violations unless they understand the SLA document. To achieve semantic interoperability and reduce the ambiguity in automating negotiation and monitoring activities, the common solution is to create SLA ontologies [19]. Current SLAs languages for cloud services and web services do not capture characteristics of IoT services. How to draft SLAs with respect to describing IoT services abstracted from the large, distributed and heterogeneous sources is still a problem.

In light of this gap, this paper presents an ontology for automatic SLA management in an IoT environment: WIoT-SLA. This ontology combines two of the most commonly-used web service SLA specifications: WS-Agreement [1] and WSLA [16]. These languages have complementary features: WS-Agreement has a well-structured schema with supports for the extension of new domain-specific elements and SLA negotiation, while WSLA defines metric descriptions of SLA parameters. This paper extend the WS-Agreement schema with a set of general IoT domain-specific concepts, which enables constraint-based SLA modeling for automatic service selection, SLA negotiation and SLA creation.

The reminder of the paper is organized as follows: Sect. 2 summarizes related work. Section 3 describes the ontology-based SLA management for IoT services. Section 4 proposes the contextual SLA ontology. Section 5 presents the SLA template match-making algorithm for optimized candidate service selection. Section 6 details the experimental setup and evaluation results and Sect. 7 concludes the paper with a discussion about future research directions.

2 Related Work

SLA specification languages are central to the definition of an SLA contract. There have been significant works in defining SLA languages for web services and cloud services. IBM published the Web Service Level Agreement (WSLA), which provides a specification for the definition and monitoring of SLAs within a web service environment [16]. WS-Agreement is another XML-based web service SLA specification defined by Open Grid Forum (OGF) [1]. Compared to WSLA, it defines a decoupled negotiation layer on top of the agreement layer for bilateral multi-round negotiation: WS-Agreement Negotiation [26]. Inspired by WS-Agreement, Uriarte *et al.* proposed an SLA language for the cloud computing domain. They predefined a set of metrics for Infrastructure-as-a-Service and adopted a denotational semantics [24]. To be decoupled from the XML-schema, the SLA@SOI project proposed an abstract SLA syntax named SLA*

[13] to automate the cloud SLA life cycle. Based on WSLA and SLA*, CSLA was proposed to address SLA violations in cloud computing [14], which supports cloud elasticity management such as the QoS or functionality degradation.

Compared to web services and cloud services, SLA specification targeting IoT services is very limited [17]. Although Gaillard *et al.* [9] extended the WSLA specification with device information to describe network performance for WSN operators, it focused on modeling the SLAs on the device layer instead of the service layer, while service consumers may be additionally interested in service-oriented aspects rather than just the concrete device information. Current research has developed a number of ontologies to model sensors and sensor observations. The SSN ontology [7] is a high-level model to describe devices' measurement capabilities and related attributes, which is further extended by OpenIoT [20] and IoT-Lite [2] to define sensors, measurements, and locations. The SENSEI project [25] models *Real World Entities* as resources, which are described by a semantic ontology including resource type, location, temporal availability, semantic operation description (e.g., input, output, pre-conditions, post-conditions), observation area, quality, and cost.

Several IoT middlewares also proposed their QoS metrics. For example, OpenIoT defines a set of utility metrics for different IoT layers to manage QoS, which includes energy consumption, delay, bandwidth, latency, etc [5]. The CityPulse project listed the quality categories that are used to assess the quality of observations made in the real world, and summarized the quality parameters with corresponding measurement units and value ranges [23]. However, as far as we know, these QoS metrics have not been integrated into the SLA schema for IoT services.

3 Ontology-Based SLA Management

From the European Commission report on recent cloud computing projects that cover SLAs, the SLA lifecycle meta-model consists of six main phases [15]: **Service use**, which reflects the information on service usage by a consumer. **Service modeling**, which deals with the service design and analysis issues, such as estimating performance and instantiating service parameters. **SLA template definition**, which creates SLA templates by analyzing the business objectives. **SLA instantiation**, which covers various processes including attributes mapping and translation, provider discovery, and dynamic SLA (re-)negotiation. **SLA enforcement**, which aims to verify the reliability of pre-negotiated QoS parameters during the service provisioning time by adopting a QoS monitoring mechanism. **SLA conclusion**, which handles the termination of signed SLAs according to pre-defined accounting and billing mechanisms. If an SLA is terminated as a violated agreement or is predicted to be violated by the QoS monitor, SLA renegotiation may be conducted as a corrective action to maintain service continuity.

The lifecycle meta-model briefly describes how to create and manage SLA-supported services. Since the IoT is a large-scale environment where multiple

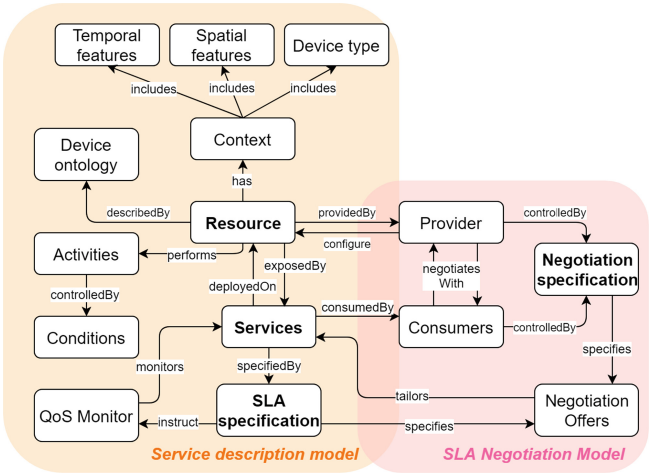


Fig. 1. Upper ontology of IoT SLA core concepts

services offering the same or similar functionalities are distributed in different locations, human intervention may not be feasible to manage services and SLAs. We assume a middleware can be deployed in different IoT gateways, which works in a distributed manner to provide the necessary functionalities such as service selection, SLA negotiation, and QoS monitoring. The service providers outline the functionalities of their SLA-supported services with default values in the form of SLA templates (SLAT) and register them to gateways so that their offerings can be discovered when requests are received by gateways from consumers. Figure 1 shows the upper ontology that describes the relations between the domain-specific core concepts of IoT services. To automate the SLA lifecycle in the IoT environment, a common global knowledge of SLAs is needed to make SLAs reciprocally understandable. This uniform SLA ontology not only allows providers to express their offerings in a standardized way but also helps gateways dynamically adjust the negotiation and monitoring mechanisms based on the metrics, constraints, and conditions defined in the SLAs. Figure 2 presents our ontology-based SLA management model: (i) SLA ontology generalizes the semantics of SLA specification and negotiation specification; (ii) The dynamic SLA negotiation and SLA creation can be performed according to the negotiation context, creation constraints and validation rules specified in the SLAT; (iii) The automatic monitoring can be conducted according to the assessment information (e.g. negotiated guarantees, measurement metrics and assessment schedulers) specified in the SLA; (iv) The service adaptation (i.e., SLA renegotiation) and accounting mechanisms can be triggered by monitored results.

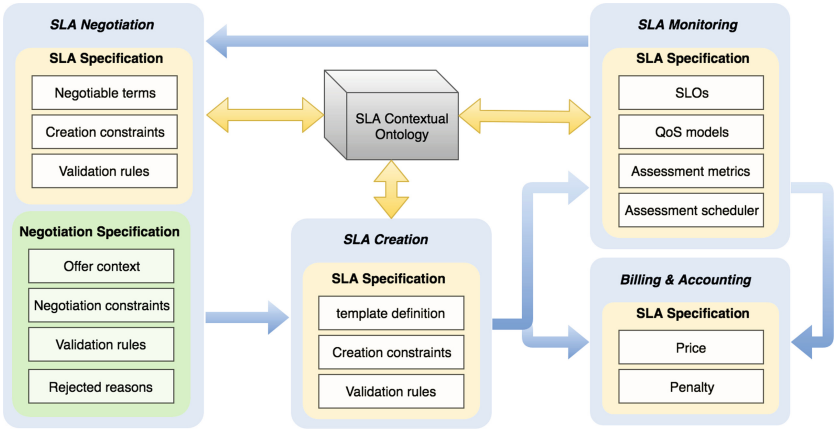


Fig. 2. Ontology-based SLA management

4 SLA Contextual Ontology

The SLA specification defines the standard format of an agreement, which requires a non-ambiguous description of services. The key requirements of defining an SLA schema in the IoT environment are simplicity, reusability, readability and efficiency [18]. In other words, the SLA ontology should contain all the necessary information for automatic SLA management but in the meanwhile, remain as simple as possible. In IoT, three well-accepted concepts are the entity, resource and service [2]. The semantic models of these concepts are associated with each other by attributes such as location, domain information, physical concept and observations. Since an end user may focus more on the service-oriented aspects rather than the physical sensor information described by a sensor ontology, it is useful to merge the important attributes of the entity model and resource model to the service model, and provide a uniform SLA ontology for IoT services.

We built an SLA ontology, called WIoT-SLA, based on an extendable web service SLA specification WS-Agreement (WSAG) [1], which supports for SLA negotiation and widely used in cloud computing projects [15]. We formalized the structure of WIoT-SLA by extending WSAG with IoT domain-specific concepts to improve the readability and efficient traversability of SLA and SLAT. The steps to achieve this were: (i) We linked the domain knowledge relating to sensing, actuating and processing tasks to real-world resources, which enables more flexible and scalable solutions for different IoT application tasks (Sect. 4.1: *Service Description Term*). (ii) We proposed a high-level abstraction of sensing service configuration properties, enabling applications to avoid complex details about devices (e.g., Fig. 5). (iii) We defined the syntax of guarantees and QoS metrics to facilitate SLA monitoring (Sect. 4.1: *Service Property*). (iv) We extended the WSAG template schema to solve the template synchronization problem and reduce the message payload during SLA negotiation (Sect. 4.1: *SLA Template Structure*). (v) We formalized the structure of negotiation offers

by extending the WS-Agreement Negotiation specification (WSAG-Negotiation) with offer validation rules and rejected reasons to avoid invalid interactions (Sect. 4.2).

4.1 SLA Description Ontology

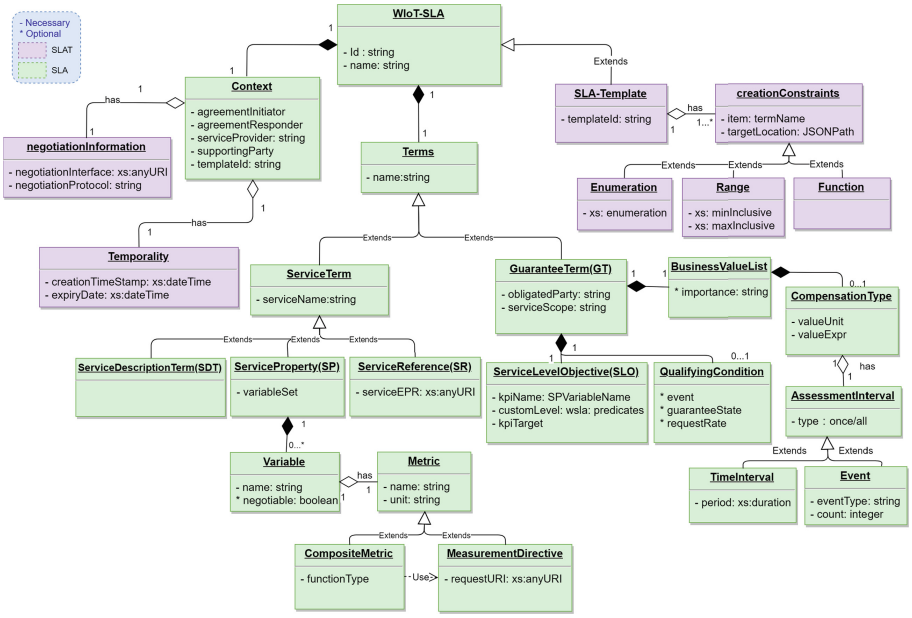


Fig. 3. WIoT-SLA structure

The structure of WIoT-SLA comprises two parts (shown in green on Fig. 3): agreement context (i.e., party information, expiry date, agreement template identifier, etc.) and terms. Services are described by *terms*, which consists of *service description term* (SDT), *service property* (SP), *service reference* (SR), and *guarantee term* (GT). SDT is the fundamental component of an SLA, which describes the functionality that will be delivered by the service. SP is used to define the measurable and exposed QoS properties associated with the service. SR (optional) lists the references point to the service (e.g., a WSDL document or a restful web service interface). GT defines the assurance of SP variables in the form of SLOs, which specifies a customized quality level that is guaranteed by the obligated party. The business objectives associated with an SLO are defined in business value list, which includes the compensation type (i.e., price and penalty) and possibly, the importance factor. The latter would be useful for SLA negotiation if the tradeoff negotiation tactic (i.e. the concession rate can be made based on the weight of each SLO) is adopted. The compensation

type specifies the consequence of SLA fulfillment, which is associated with an assessment interval describing how to measure the SLA violation for monitors (e.g., periodic schedule or the number of events that has occurred).

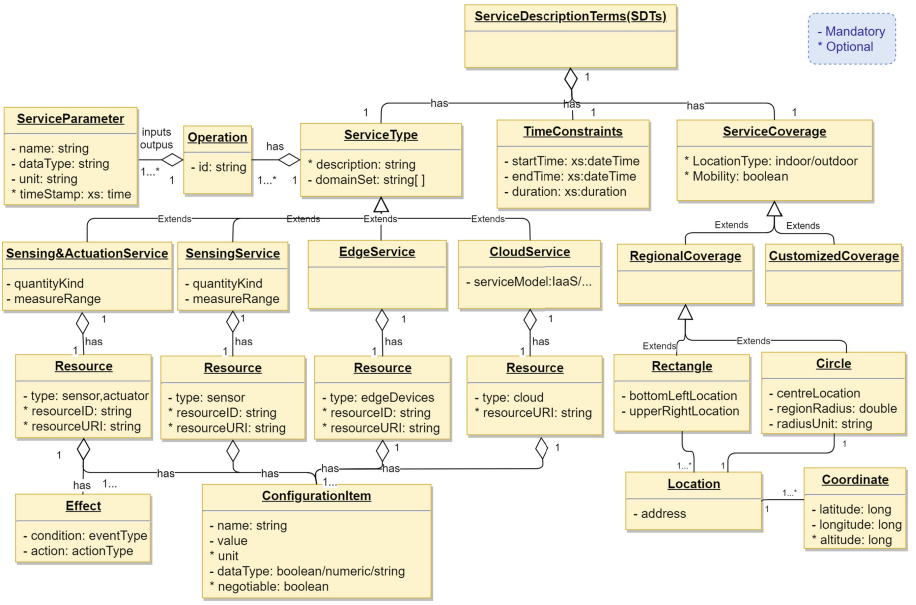


Fig. 4. Service description terms

Service Description Term. Figure 4 shows the ontology of service description terms in WIoT-SLA, which consists of three parts: time constraints, service coverage, and service type. **Time constraints** specifies the actual service provisioning time, which is different from the temporality in an SLAT. **Service coverage** specifies the spatial features of the service (e.g., the observation area of sensors). We pre-defined the rectangle area and circle area for regional coverage. The concrete location can be defined with an address ontology or the geographic coordinate.

Service type generalizes a service's functionality with domain information, operation, service parameters (i.e., input and output) and configurable features. In IoT, the service type can be clustered as a sensing service (e.g., temperature sensing), a sensing and actuation service (e.g., trigger the alarm when detected hazard gas concentration greater than a threshold), an edge service (e.g., a data dispatch service that collects data from sensors and publish verified data to subscribed services) and cloud service (e.g., data storage and data processing service that analyzing historical data and predict abnormalities) based on the

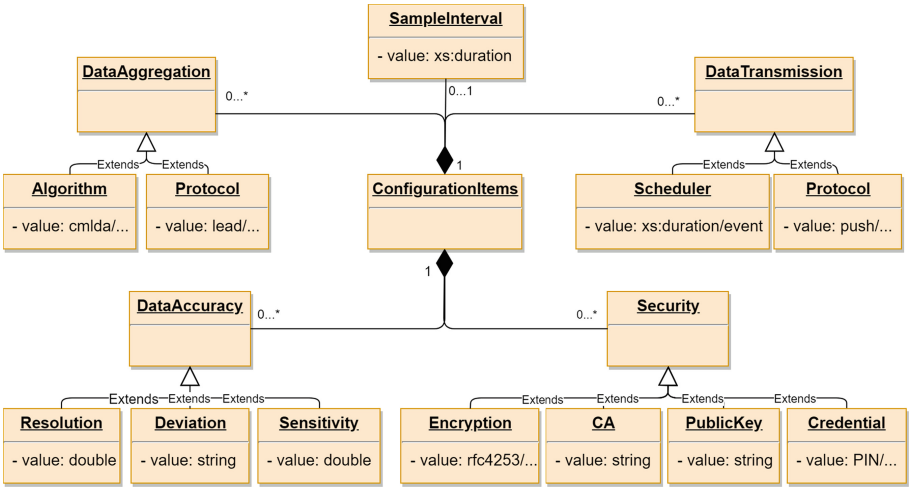


Fig. 5. An example of configuration items (sensing service)

deployed resources. For instance, quality kind can be used to describe the real-world property observed by the sensors, but a sensing and actuation service may have an "effect" attribute describing an event or an action when the pre-defined condition is met. Each resource type is associated with one or more configuration items, which specifies the service's functional features, such as the sample rate for sensing services, data reporting rate for edge services or memory capacity for cloud services. The configuration item is defined with a name, a value, and data type (i.e., boolean, numeric and string) of the value. Figure 5 outlined a set of general configuration items for sensing services, including data accuracy, security, sample rate, data aggregation, and data transmission.

Service Property. In WIoT-SLA, SP variables are defined as the dynamic QoS features (i.e., values are affected by devices' status and run-time environment) that can be monitored by a measurement party. Each SP variable is described by a name, which is further used in expressing SLO in GTs (i.e., *KPIName* in SLO), and the customized metrics specifying its measurement unit. Different from cloud computing and web services, the QoS model in the IoT context needs to consider the complexity introduced by the layered architecture of IoT applications [8]. From bottom to top, the architecture is composed of a perception layer, network layer, service layer, and application layer. In the **perception layer**, a heterogeneous set of devices sample the state of the physical world with different capabilities and constraints. Quality metrics for a perception layer include data correctness, data completeness, transmission speed, energy consumption, price, etc [23]. The **network layer** comprises the network infrastructures on which the IoT platform is based. Quality metrics for the network layer focus on the performance of data transmissions, such as network delay, bandwidth,

packet loss, and network jitter [6]. The **service layer** processes information received from the lower layer and provides services such as data storage, data management, and data analysis. *CLOUDQUAL* is an example of a quality model for cloud services, which defines usability, availability, reliability, responsiveness, security, and elasticity as quality dimensions [27]. In the **application layer**, different services provided by the lower layer can be composed together to fulfill the domain-specific requirements of an application task. The end-to-end QoS of the application layer is dependent on the aggregated or nested QoS metrics across the lower layers. For instance, a fire detection application has stringent QoS demands on availability, responsiveness and accuracy, which constrains the quality level of each layer, such as good quality sensors with high data precision, adjustable sampling rate, fast transmission speed, available networks with low latency, and a high reliability for the data processing service. Since achieving SLOs at the application layer requires the satisfaction of the SLOs of lower level services, the guarantee states[1] of lower-level service properties can be used as a precondition under which the application SLOs take effect. The precondition can be specified in the *QualifyingCondition* associated with each SLO.

Since the semantics of QoS parameters are not defined precisely in WSAG, we define two types of metrics in WIoT-SLA: measurement directives and composite metrics. The measurement directive is derived from WSLA specification [16], and is directly retrieved from managed resources (e.g., a request URI exposed by the QoS monitor). The composite metrics are created by aggregating measurement directives or other composite metrics according to a function. For example, availability is a composite metric that is composed by a *measurementDirective* (i.e., service uptime and service execution time) with a function (i.e., the ratio of the service uptime to the service execution time).

SLA Template Structure. SLAT is designed as a blueprint to create a valid SLA and SLA negotiation offer, which shares the same structure as the final SLA except for some additional segments. In WIoT-SLA, these segments are (marked in purple on Fig. 3): *CreationConstraint* (optional), *Temporality* (mandatory), *negotiationInformation* (optional) and *Negotiable* indicator (optional) for configuration items or SP variables. The items specified in *CreationConstraint* must be presented in a valid initial negotiation offer and the final SLA with the values satisfying the constraints. We divide constraints into three types: *Range*, *Enumeration* and *Function*. The *Range* type specifies the minimum and maximum value, the *Enumeration* type lists all the possible values, and the *Function* type specifies the value in the form of a function. We define constraints with two attributes: *item* and *targetLocation*, which specifies the name of a term and where to put the constraint respectively. The *targetLocation* can be expressed using a querying languages such as JSONPath[2]. The *Temporality* is defined to indicate the validity date of an SLAT, which is composed of creation timestamp and expiry date. Since the IoT is a large-scale distributed environment

[1] WSAG guarantee state model represents a fulfillment state for each GT of an SLA.
[2] https://goessner.net/articles/JsonPath/index.html#e2 - Accessed 15 Jan 2019.

and providers' offerings may change as time passes, the creation timestamp is used to synchronize the latest version of SLAT within the system. The expiry date is used to allow IoT middlewares periodically check the availability of registered SLATs and remove the expired ones to avoid unnecessary interactions during the negotiation stage. The *negotiationInformation* specifies the negotiation interface (i.e., a restful negotiation service EPR or an instant message address) and the negotiation protocol (e.g., CNP or WSAG-Negotiation) if the service is negotiable. If an SLAT does not specify this segment, the service is non-negotiable, and users have to accept all the default values specified in the SLAT. The *Negotiable* indicator is defined to specify the negotiable terms whose values can be changed through negotiation, which means the value in the final SLA can be different from the default value specified in the SLAT. If the indicator is omitted, this means the term is non-negotiable, and it must hold the default value presented in the SLAT.

4.2 Negotiation Offer Ontology

WSAG-Negotiation formalizes negotiation information as *negotiation offers* [26], which are generated based on an SLAT. Generally, the ontology of a negotiation offer (Fig. 6) has four sections: negotiation context (e.g., identifier, party, etc.), offer context, negotiable terms, and negotiation constraints. A negotiation constraint specifies the constraints on negotiable terms when creating a valid counteroffer, which has a similar format to a creation constraint, except that an additional constraint type *FixedValues* is added to indicate the value can not be changed in subsequent offers or the final SLA. The offer state model specified in an offer context controls the interactions between negotiation parties and indicates the rules for taking action after receiving a new offer. For offers in the "rejected" state, to reduce ambiguity and avoid futile interactions, the offer context is extended with domain-specific information to indicate why the offer is rejected. The set of predefined rejected reasons are: {*UnsupportedTerm*, *SLOConflict*, *Timeout*, *InvalidOffer*, *UnderPayment*}. Considering the negotiation offer specified by WSAG-Negotiation might be too heavyweight during the negotiation process, we regulate that the SLAT must be referred in each negotiation offer, and only the terms that specified in *CreationConstraints* or have *Negotiable* indicators will be presented in negotiation offers. Other terms are omitted but regarded as holding the same values presented in the SLAT. Any inconsistency will cause a failure when validating the negotiation offer or creating the final agreement.

5 SLA Template Match-Making

As we described in Sect. 3, SLA negotiation is the first and necessary step to create an SLA before the actual service delivery. Considering the scale of IoT services and possible long latency during a bilateral negotiation process, a template match-making process can reduce the negotiation time by ranking the

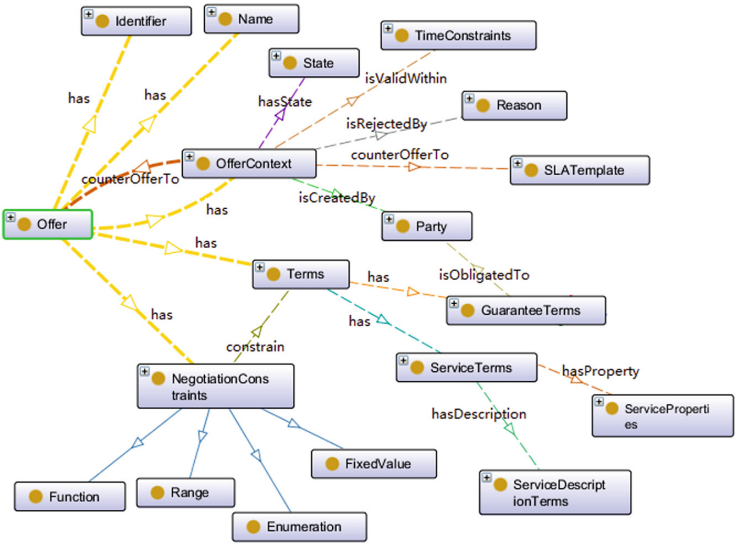


Fig. 6. Ontology of negotiation offer

candidate services based on the similarity between requests and services. Gateways associate an incoming request with registered SLATs that are closest to the requirements, and select the most optimized SLAT to create a negotiation offer. A request is defined as $R = \langle I, O, T, L, F, Q, C^* \rangle$, consisting of inputs, outputs, time, location, functional features (e.g., sample interval), QoS properties, and possibly, the constraints (e.g., the expected price range). We assume providers formalize SLATs by following the WIoT-SLA structure, and the service discovery engine uses a goal-driven backward planning algorithm to discover candidate services based on the semantic relations of service parameters [3]. The matchmaking process is composed of two steps: (i) The candidate SLATs are filtered based on time and spatial features. (ii) The correspondence between the request and each candidate SLAT is measured by the similarity of the request and the SLAT, which is the weighted sum of semantic similarities between the requested terms and the offered terms.

To calculate the semantic similarity of requested terms and services terms, we use an auxiliary source *WordNet* to compute the *WUP* relatedness [11]. A score value greater than 0.75 is regarded as a valid matching. The weight of each valid matching is calculated based on the data type, default value and creation constraints presented in the SLAT. If the feature is negotiable without any creation constraints, the weight w is set to 1. If the feature is negotiable and the value is restricted in the constraints, the weight of a string type is measured by the minimum *Levenshtein distance* [10], and the weight of a numeric type is calculated by Eq. 1 for lower-is-better features (e.g., price or sample interval from a consumer's perspective), and by Eq. 2 for higher-is-better features (e.g.,

availability or reliability from a consumer's perspective).

$$w = \begin{cases} 1, & \text{if } S_{max} \leq R_{min} \\ \frac{|R \cap S|}{|R|}, & \text{otherwise.} \end{cases} \quad (1)$$

$$w = \begin{cases} 1, & \text{if } S_{min} \geq R_{max} \\ \frac{|R \cap S|}{|R|}, & \text{otherwise.} \end{cases} \quad (2)$$

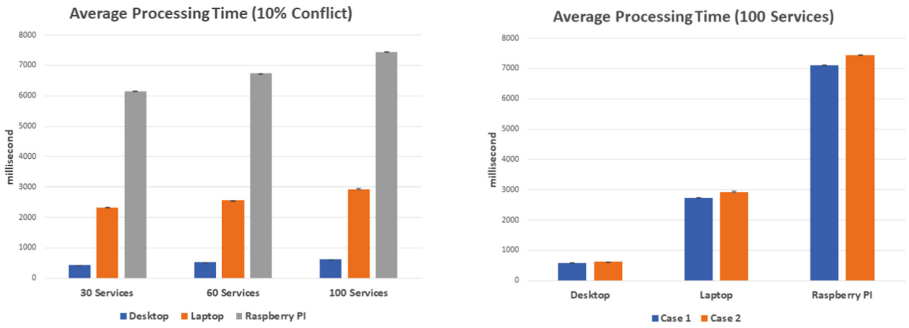
where S_{max}, S_{min}, R_{max}, R_{max} are the maximum and minimum values of the offered feature and the requested feature respectively. $R \cap S$ is the intersection between the request values and the offered values.

6 Evaluation

A user requests a hazardous gas detection service with functional requirements including minimum sample interval, maximum data deviation, access credential and data reporting protocol, and the QoS requirements including price, availability, reliability and latency. Based on examples proposed in IoT literature [4], we established a SLAT prototype of a gas detection service, and created three datasets based on the prototype. In the first dataset (i.e., test case 1), only 20% services match the request, while in the second dataset (i.e., test case 2), the percentage is increased to 90% (i.e., 10% conflict). For the services that violate the request, 50% of them conflict with the spatial requirements and the rest conflict with the functional or QoS requirements. In the third dataset (i.e., test case 3), 60% services match the request and 60% of them present the same service properties using different words (i.e., using robustness to represent reliability), 40% of them adopt different names as well as data types. In each dataset, 30, 60 and 100 JSON-formatted SLATs are created according to the structure of WIoT-SLA. Theses SLATs are different in terms of configuration items (i.e., different names, synonymous names, different range of values, different data types), SP variables, SLOs and constraints.

The WIoT-SLA match-making algorithm is implemented in Java under Eclipse Mars2 IDE, and the third-party library WS4J[3] is integrated to check words' semantic relatedness. The executable jar file is deployed on three devices: a Dell-OptiPlex-990 desktop (Intel Core i7-2600 CPU, 4 GB DDR3 1333 MHz RAM, Windows 10 OS), a 13-inch MacBook-Pro laptop (Intel Core i5 CPU, 8 GB DDR3 1333 MHz RAM, macOS High Sierra) and a Raspberry Pi-3 (4xCortex-A7 CPU, 1 GB RAM, 16 GB SD card, Raspbian OS). Figure 7 shows the average processing time (APT) on each device as the scale of candidate services increases, and the APT under the first two different test cases respectively. The service scale has a negative impact on the performance of our SLA match-making algorithm (7(a)), but the negative impact can be slightly reduced by adopting location-based filtering (7(b)). From the result, the responsiveness of WIoT-SLA

[3] https://code.google.com/archive/p/ws4j/ - Accessed 22 Jan 2019.



(a) APT of different service scale (b) APT of different test cases

Fig. 7. Average processing time

match-making is highly dependent on gateways' computational capabilities. For instance, the APT on Dell-OptiPlex-990 (approximately 622 ms) is about 12 times of that on Raspberry Pi-3 (approximately 7438 ms). If we assume gateways are the resource-rich edge devices that can manage hundreds of services that deployed in the local area, such as a desktop or a personal workstation, the latency is acceptable. Otherwise, a more light-weight SLA match-making algorithm is needed for resource-constrained devices selecting the services that have a bigger chance to satisfy all the requirements through SLA negotiation. We further compute the average precision, recall and accuracy in template march-making for test case 3, and compare the result with path-length based similarity (PATH) and Lin similarity (LIN) [11]. Figure 8 presents the result under different thresholds. Among these three approaches, WUP shows a better and more stable performance in a wider range, that's why we select the WUP similarity and set the threshold to 0.75. Although there are services incorrectly matched to the request, considering the high recall, this problem can be solved by ranking the candidate services based on their similarity value and selecting the Top-K solutions as the final candidate services.

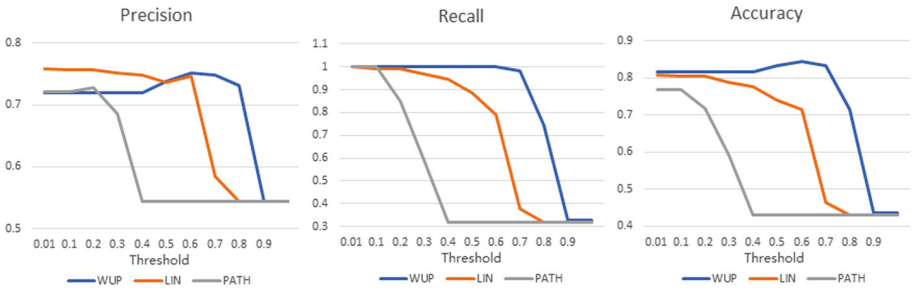


Fig. 8. Precision, recall and accuracy of test case 3

7 Conclusion

This paper proposed an SLA ontology for IoT services that covers the IoT's layered architecture and its domain-specific properties. To achieve semantic interoperability and enable automatic SLA management in the IoT environment, we formalized the SLA schema by extending the commonly used web service SLA specification: WS-Agreement. This schema can be extended by domain-specific experts to construct SLAs for different applications. According to the ontology, we designed a match-making algorithm to select the candidate services which are more likely to provide the service as requested before SLA negotiation. As future work, we aim to develop an SLA reputation system that can audit the SLA's fulfillment based on negotiation and monitoring result, and provide a more lightweight service match-making mechanism resource-constrained IoT gateways.

References

1. Andrieux, A., et al.: Web services agreement specification (WS-agreement). In: Open Grid Forum, vol. 128, p. 216 (2007)
2. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: IoT-Lite: a lightweight semantic model for the Internet of Things. In: 2016 International IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), pp. 90–97. IEEE (2016)
3. Cabrera, C., Palade, A., White, G., Clarke, S.: Services in IoT: a service planning model based on consumer feedback. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) ICSOC 2018. LNCS, vol. 11236, pp. 304–313. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_21
4. Cabrera, C., White, G., Palade, A., Clarke, S.: The right service at the right place: a service model for smart cities. In: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 1–10. IEEE (2018)
5. Calbimonte, J.P., Riahi, M., Kefalakis, N., Soldatos, J., Zaslavsky, A.: Utility metrics specifications. openiot deliverable d422. Technical report (2014)
6. Chen, D., Varshney, P.K.: QoS support in wireless sensor networks: a survey. In: International Conference on Wireless Networks, vol. 233, pp. 1–7 (2004)
7. Compton, M., et al.: The SSN ontology of the W3C semantic sensor network incubator group. Web Semant.: Sci. Serv. Agents World Wide Web **17**, 25–32 (2012). https://doi.org/10.1016/j.websem.2012.05.003. http://www.sciencedirect.com/science/article/pii/S1570826812000571
8. Duan, R., Chen, X., Xing, T.: A QoS architecture for IoT. In: 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing Internet of Things (iThings/CPSCom), pp. 717–720. IEEE (2011)
9. Gaillard, G., Barthel, D., Theoleyre, F., Valois, F.: SLA Specification for IoT Operation-The WSN-SLA Framework. Ph.D. thesis, INRIA (2014)
10. Gusfield, D.: Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1997)
11. Jurafsky, D., Martin, J.H.: Speech and Language Processing, vol. 3. Pearson, London (2014)

12. Kazmi, A., Serrano, M., Lenis, A., Soldatos, J.: A QoS-aware integrated management of IoT deployments in smart cities. In: 2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA), pp. 141–146. IEEE (2017)
13. Kearney, K.T., Torelli, F., Kotsokalis, C.: SLA*: an abstract syntax for service level agreements. In: 2010 11th IEEE/ACM International Conference on Grid Computing (GRID), pp. 217–224. IEEE (2010)
14. Kouki, Y., De Oliveira, F.A., Dupont, S., Ledoux, T.: A language support for cloud elasticity management. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 206–215. IEEE (2014)
15. Kyriazis, D.: Cloud computing service level agreements-exploitation of research results. European Commission Directorate General Communications Networks Content and Technology Unit, Technical report 5, 29 (2013)
16. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web service level agreement (WSLA) language specification, pp. 815–824. IBM Corporation (2003)
17. Palade, A., Cabrera, C., Li, F., White, G., Razzaque, M., Clarke, S.: Middleware for Internet of Things: an evaluation in a small-scale IoT environment. J. Reliable Intell. Environ. **4**, 1–21 (2018)
18. Papadopoulos, A.V., Asadollah, S.A., Ashjaei, M., Mubeen, S., Pei-Breivold, H., Behnam, M.: SLAs for industrial IoT: mind the gap. In: 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp. 75–78. IEEE (2017)
19. Redl, C., Breskovic, I., Brandic, I., Dustdar, S.: Automatic SLA matching and provider selection in grid and cloud computing markets. In: Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, pp. 85–94. IEEE Computer Society (2012)
20. Soldatos, J., et al.: OpenIoT: open source Internet-of-Things in the cloud. In: Podnar Žarko, I., Pripužić, K., Serrano, M. (eds.) Interoperability and Open-Source Solutions for the Internet of Things. LNCS, vol. 9001, pp. 13–25. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16546-2_3
21. Swiatek, P., Rucinski, A.: IoT as a service system for eHealth. In: 2013 IEEE 15th International Conference on e-Health Networking, Applications & Services (Healthcom), pp. 81–84. IEEE (2013)
22. Thoma, M., Meyer, S., Sperner, K., Meissner, S., Braun, T.: On IoT-services: survey, classification and enterprise integration. In: 2012 IEEE International Conference on Green Computing and Communications (GreenCom), pp. 257–260. IEEE (2012)
23. Tönjes, R., et al.: Real time IoT stream processing and large-scale data analytics for smart city applications. In: Poster session, European Conference on Networks and Communications (2014)
24. Uriarte, R.B., Tiezzi, F., Nicola, R.D.: SLAC: a formal service-level-agreement language for cloud computing. In: Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 419–426. IEEE Computer Society (2014)
25. Villalonga, C., Bauer, M., López Aguilar, F., Huang, V.A., Strohbach, M.: A resource model for the real world internet. In: Lukowicz, P., Kunze, K., Kortuem, G. (eds.) EuroSSC 2010. LNCS, vol. 6446, pp. 163–176. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16982-3_13
26. Waeldrich, O., et al.: WS-Agreement Negotiation Version 1.0, p. 64 (2011)
27. Zheng, X.: QoS representation, negotiation and assurance in cloud services. Queen's University (Canada) (2014)