Noname manuscript No.
(will be inserted by the editor)

Towards a Quality-of-Thing based Approach for Assigning Things to Federations

Zakaria Maamar · Muhammad Asim · Khouloud Boukadi · Thar Baker · Saad Saeed · Ikbel Guidara · Fadwa Yahya · Emir Ugljanin · Djamal Benslimane

Received: date / Accepted: date

Abstract In the context of an Internet-of-Things (IoT) ecosystem, this paper discusses 2 necessary stages for managing federations of things. The first stage defines things in terms of duties and non-functional properties that define the quality of these duties. And, the second stage uses these properties to assign appropriate things to future federations. Specialized into adhoc and planned, federations are expected to satisfy needs and requirements of real-life situations like traffic control that arise at run-time. A set of experiments using a mix of real and simulated datasets, demonstrate the technical doability of thing assignment to federations and are presented in the paper, as well.

Keywords Federation · IoT · Quality-of-Things · Assignment.

1 Introduction

In [10], we presented the concept of Thing-Federation-as-a-Service (TFaaS) as a novel way to address the *silo* constraint that impedes the collaboration of IoT-compliant things (things, for short). A federation is a group of things

Z. Maamar
Zayed University, Dubai, UAE
E-mail: zakaria.maamar@zu.ac.ae

M. Asim and S. Saeed
FAST-NUCES, Islamabad, Pakistan

K. Boukadi and F. Yahya
University of Sfax, Sfax, Tunisia

T. Baker
Liverpool John Moores University, Liverpool, UK

I. Guidara and D. Benslimane
Claude Bernard Lyon 1 University, Lyon, France

E. Ugljanin
State University of Novi Pazar: Novi Pazar, Serbia

that are put together in order to handle a particular real-life situation such as *tunnel closure* triggered by a car accident. With respect to this situation, things forming a federation could be traffic light, speed-limit sign, speed camera, and flip-disc display; they all collaborate towards achieving smooth traffic diversion. We also presented in [10] how things are defined using a set of Quality-of-Thing (QoT) non-functional properties, how federations are specialized into planned and ad-hoc, and how federations could be deployed on top of a cloud/edge architecture. In conjunction with these contributions, a set of experiments using emergency services were carried out to demonstrate the technical doability of thing federation. In this paper that extends our work in [10], we focus on QoT-based assignment of things to federations.

According to Gartner[1], 6.4 billion connected things were in use in 2016, up 3% from 2015, and will reach 20.8 billion by 2020. The wireless world research forum also reported that in 2017, there were 7 trillion wireless devices serving 7 billion people leading to IoT formation [16]. This large and ever-growing number of things need to be "harnessed" so, that, things' collective over individual behaviors prevail. We exemplify the collective behavior by combining things' duties that we specialize into sensing, reasoning, actuating, and communicating [9]. In fact, a federation is about combining separate duties from separate things.

To respond to changes in IoT ecosystems, we specialize federation into planned and ad-hoc. On the one hand, the former is put in place ahead of time and, hence, has its thing members already identified with respect to a situation's needs and requirements. On the other hand, the latter is put in place on-the-fly when none of the ready-to-use/available planned federations can handle a situation and, hence, needs to have its thing members identified, selected, and finally assigned. According to Perera et al. [2], the existence of multiple things will facilitate the emerge of marketplaces of things. Tapping into these marketplaces' opportunities (e.g., competition and substitution between things) would require new IoT-based mechanisms and means for defining, announcing, and selecting things. To address the concern of thing selection, we resort to existing practices like those based on Quality-of-Service (QoS) [3,8] to develop our thing selection approach based on QoT properties. Our QoT properties permit to capture things' peculiarities in terms of reduced size, restricted connectivity, continuous mobility, limited energy, and constrained storage.

Our contributions compared to [10] include adoption of QoT model when selecting things, development of a thing selection process, and implementation of the process using a mix of real and simulated datasets of things. The rest of this paper is organized as follows. Section 2 presents the QoT model. Section 3 is about thing federation in terms of concepts and operations. Section 4 presents the thing selection process that leads to assigning things to federations. Section 5 is an overview of some related work. Finally, Section 6 concludes the paper and presents some future work.

[1] `www.gartner.com/newsroom/id/3165317`.

2 Quality-of-Things model

By defining a QoT model for things we align ourselves with the trend of developing similar models in other ICT domains such as quality model for cloud services [4] and quality model for Web services [3]. To enable a competitive selection of things with respect to a situation's requirements, we resort to our QoT model that captures in an abstract way the 4 core duties of a thing that are sensing, reasoning, actuating, and communicating [10]. In Fig. 1, (0,1) refers to either disabling or enabling a duty with respect to a situation's requirements (e.g., traffic-status broadcasting does not require reasoning) and filled circles identify the duties that interact with the cyber-physical surrounding. Fig. 2 is a screenshot of our in-house tool allowing to define things' duties.

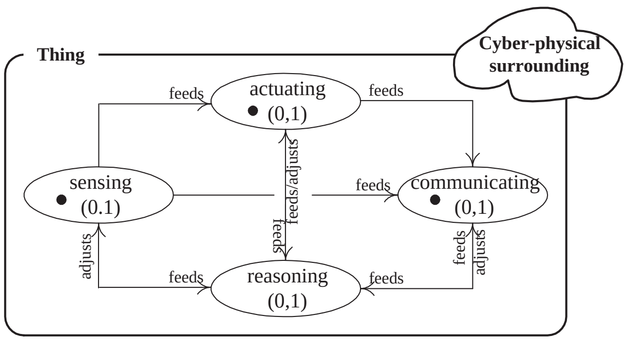
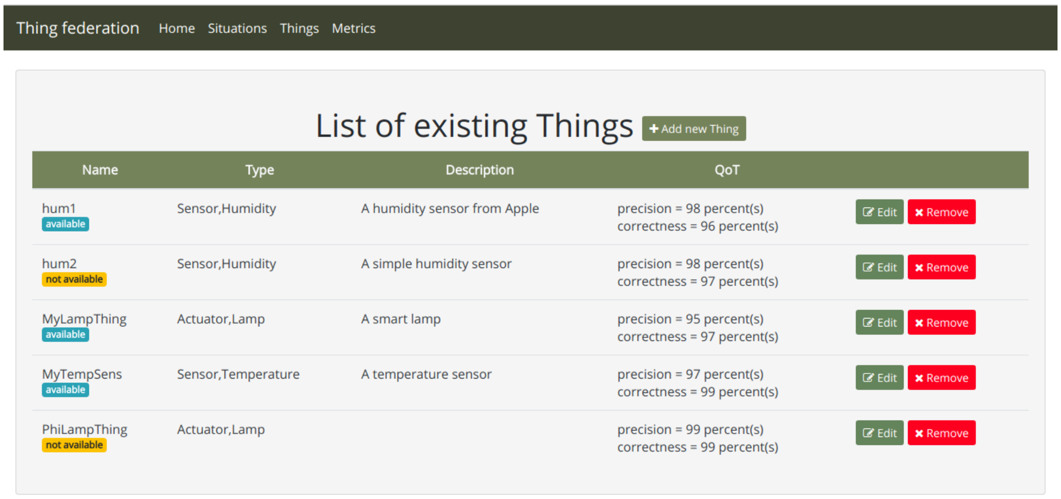


Fig. 1 Duties defining a thing's QoT model

From an operational perspective, a thing senses the cyber-physical surrounding to generate (raw) data that would be shared with actuating, reasoning, and/or communicating ("feeds" in Fig. 1); a thing actuates on top of the cyber-physical surrounding based on the sensed data and/or the data that results from reasoning; a thing reasons over the sensed and/or actuated data to make decisions that could lead to adjusting sensing, actuating, and/or communicating ("adjusts" in Fig. 1); and, finally, a thing communicates with the cyber-physical surrounding the data that result from sensing, actuating, and/or reasoning. We provide hereafter some potential QoT properties for each duty. QoT properties for sensing include:

- *Frequency of sensing* (e.g., continuous *versus* intermittent).
- *Quality of sensed outcome* that determines for instance, the accuracy and validity of the outcome (e.g., high *versus* low accuracy; high-accuracy outcome would not require any further verification).
- *Resource* (e.g., energy, CPU, and storage) consumption during sensing (e.g., high *versus* low energy).

QoT properties for actuating include:



List of existing Things

Name	Type	Description	QoT	
hum1 available	Sensor,Humidity	A humidity sensor from Apple	precision = 98 percent(s) correctness = 96 percent(s)	Edit Remove
hum2 not available	Sensor,Humidity	A simple humidity sensor	precision = 98 percent(s) correctness = 97 percent(s)	Edit Remove
MyLampThing available	Actuator,Lamp	A smart lamp	precision = 95 percent(s) correctness = 97 percent(s)	Edit Remove
MyTempSens available	Sensor,Temperature	A temperature sensor	precision = 97 percent(s) correctness = 99 percent(s)	Edit Remove
PhiLampThing not available	Actuator,Lamp		precision = 99 percent(s) correctness = 99 percent(s)	Edit Remove

Fig. 2 Tool for defining things' duties

– *Quality of actuated outcome* that determines for instance, the accuracy and validity of the outcome.
– *Resource* (e.g., energy, CPU, and storage) consumption during actuating (e.g., high *versus* low energy).

QoT properties for reasoning include:

– *Timeliness* determines the time spent processing for making a decision.
– *Accuracy* determines the decision suitability with respect to a situation's requirements.
– *Resource* (e.g., energy, CPU, and storage) consumption during reasoning (e.g., high *versus* low energy).

Finally, QoT properties for communicating include:

– *Reception rate of sensed, reasoned upon, and/or actuated outcome* (incoming flow) that determines for instance, data loss, data volume with respect to a bandwidth, etc.
  – *Acceptance rate of received outcome* is about the outcome that has been accepted for distribution; some received outcome could be rejected.
– *Delivery rate of sensed and/or actuated outcome* (outgoing flow) that determines data loss, data volume with respect to a bandwidth, etc.
  – *Acceptance rate of delivered outcome* is about the outcome that has been accepted after distribution at the recipient end; some delivered outcome could be rejected.
– *Resource* (e.g., energy and bandwidth) consumption during communicating (e.g., high *versus* low bandwidth).

In conjunction with defining the QoT model, we describe things in compliance with the Web of Things (WoT) Thing Description[2]. This description includes QoT properties per duty and other properties related to semantic metadata, security, communication, and interaction resources. For illustration purposes, Listing 1 presents a moisture-sensor description in JSON-LD. In this listing, lines 2-3 refer to semantic metadata, lines 4-6 refer to details about the thing, lines 7-17 refer to interaction resources, lines 18-21 refer to communication, lines 22-26 refer to security, and, finally, lines 27-57 refer to QoT properties.

Listing 1 Moisture sensor's WoT CP description

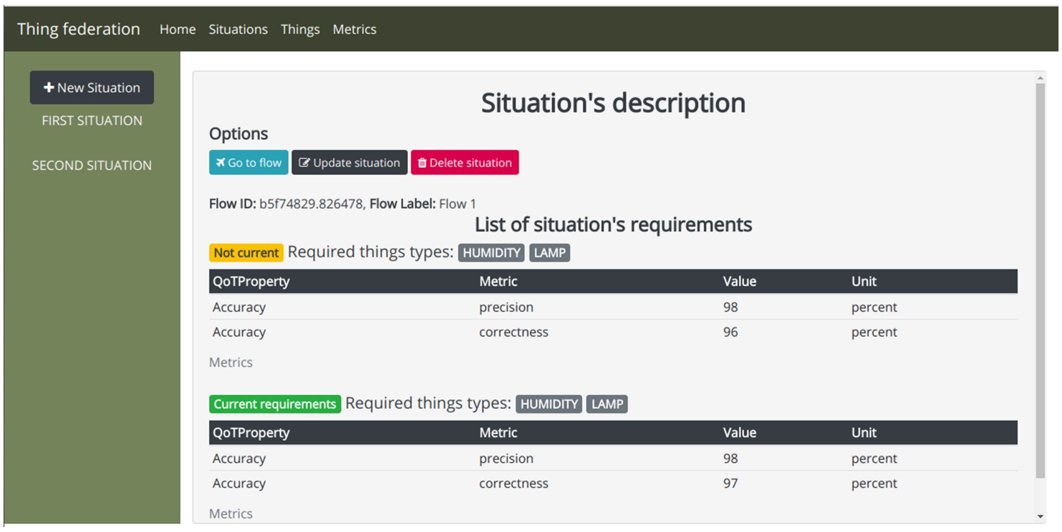
```
 1 {
 2         "@context": ["https://w3c.github.io/wot/w3c-wot-td-context.jsonld",
 3         "https://w3c.github.io/wot/w3c-wot-common-context.jsonld"],
 4         "@type": ["Sensor"],
 5         "name": "myMoistureSensor",
 6         "base": "coap:///www.example.com:5683/moisture-sensor/",
 7         "interaction": [{
 8                 "@type": ["Property","Moisture"],
 9                 "name": "MoistureSensor",
10                 "schema":{ "type": "number" },
11                 "writable": false,
12                 "observable":true,
13                 "form": [{
14                         "href": "val",
15                         "mediaType": "application/json"
16                         }],
17                 }],
18         "link": [{
19                 "href": "coap://moisture.example.com:5683/ev",
20                 "mediaType": "application/json"
21         }],
22         "security": {
23                 "cat": "token:jwt",
24                 "alg": "HS256",
25                 "as": "https://authority-issuing.example.org"
26         },
27         "quality":[{
28                         "type":"sensing",
29                         "name":"Frequency of sensing",
30                         "property":"frequency",
31                         "value":"continuous"
32                 },{
33                         "type":"sensing",
34                         "name":"Quality of sensed outcome",
35                         "property":"outcomequality",
36                         "value":"high"
37                 },{
38                         "type":"sensing",
39                         "name":"Resource",
40                         "property":["resource","energy"],
41                         "value":"high"
42                 },{
43                         "type":"communicating",
44                         "name":"Reception rate of sensed outcome",
45                         "property":["reception","bandwidth"],
46                         "value":"high"
47                 },{
48                         "type":"communicating",
49                         "name":"Delivery rate of sensed outcome",
50                         "property":["deliveryrate","bandwidth"],
```

[2] www.w3.org/TR/wot-thing-description.

```
                "value":"high"
        },{
                "type":"reasoning",
                "name":"Accuracy of decision",
                "property":"accuracy",
                "value":"high"
        }]
}
```

3 Federations of things

Federation existence strictly depends on situations that arise at run-time and hence, need to be handled (Fig. 3). Things that populate federations are expected to satisfy situations' changing requirements. With respect to the tunnel closure, on one occasion the requirements revolve around streaming quality, and on another occasion the requirements revolve around streaming reliability. Thus, it is not necessary that same things populate all federations associated with the same situation. In this part of the paper we discuss some concepts and operations related to federations and then, illustrate how federation could be reused and compared. More details about federations are included in [10].



Flow ID: b5f74829.826478, Flow Label: Flow 1

List of situation's requirements

Not current Required things types: HUMIDITY LAMP

QoTProperty	Metric	Value	Unit
Accuracy	precision	98	percent
Accuracy	correctness	96	percent

Metrics

Current requirements Required things types: HUMIDITY LAMP

QoTProperty	Metric	Value	Unit
Accuracy	precision	98	percent
Accuracy	correctness	97	percent

Metrics

Fig. 3 Tool for describing situations

3.1 Concepts

A federation gets (most likely heterogeneous) things together based on the fact that their respective QoT-based duties allow to satisfy the needs/requirements of a situation (e.g., tunnel closure) that is assigned to this federation. To differentiate planned from ad-hoc federations (Table 1), we specialize things into abstract and concrete.

– A planned federation has, at both design- and run-time, all its concrete thing members identified and ready to act, should this federation be assigned a situation.
– An ad-hoc federation has, at design-time, its abstract thing members defined. When a situation arises at run-time and none of the existing planned federations can handle this situation, concrete things instantiate the ad-hoc federation's abstract things and then, start acting. Following successful handling of the situation, we label the ad-hoc federation as planned. How to select concrete things among a set of similar things and assign the selected things to federations is detailed in Section 4.

Table 1 Planned federation *versus* Ad-hoc federation

	Types of federation	
	Planned	Ad-hoc
Design time	Necessary concrete things are QoT-based identified	Necessary abstract things are QoT-based defined
Run time	Concrete things are invoked	Abstract things are instantiated after QoT-based identification of concrete things Concrete things are invoked

To remain competitive, a federation could make a concrete thing sign-off (or eject it) if its QoT-driven performance (e.g., unreliable data and recurrent failure) does not meet this federation's expectations. We recall that federations are expected to satisfy situations' requirements. A thing can also leave a federation, should the business in the federation become less rewarding (e.g., data-sharing rate among the members drops below a threshold). To avoid/reduce departures from federations, incentives (monetary or in-kind) could be used to ensure the long-term commitment of things to federations.

Whether planned or ad-hoc, a language to define the individual and collective operations of things in federations is deemed necessary. Individual operations target separate things whereas collective operations require composing things. By analogy to other domains like Web services, composition is exemplified with either orchestration (centralized) or choreography (decentralized) [14]. In the field of IoT, Thuluva et al. propose *Recipes* as a programming language to specify the structure and configuration of IoT systems [18, 19]. Acting as an abstract template, a recipe is about ingredients and interactions that define the data flow between IoT devices, referred to as offering. A recipe's run-time instantiation happens through some offering selection rules. In another work, Khanda et al. suggest Jolie as a programming language for connecting microservice-based IoT in the context of smart buildings [7]. Jolie treats microservices as first-class citizens that can be reused, orchestrated, and aggregated with others.

3.2 Operations

Fig. 4 represents our ecosystem of both things and federations of things. Federations transition through 4 stages: assembling, storage, **either** activation (for planned federation) **or** instantiation (for ad-hoc federation), and ultimately disassembling.

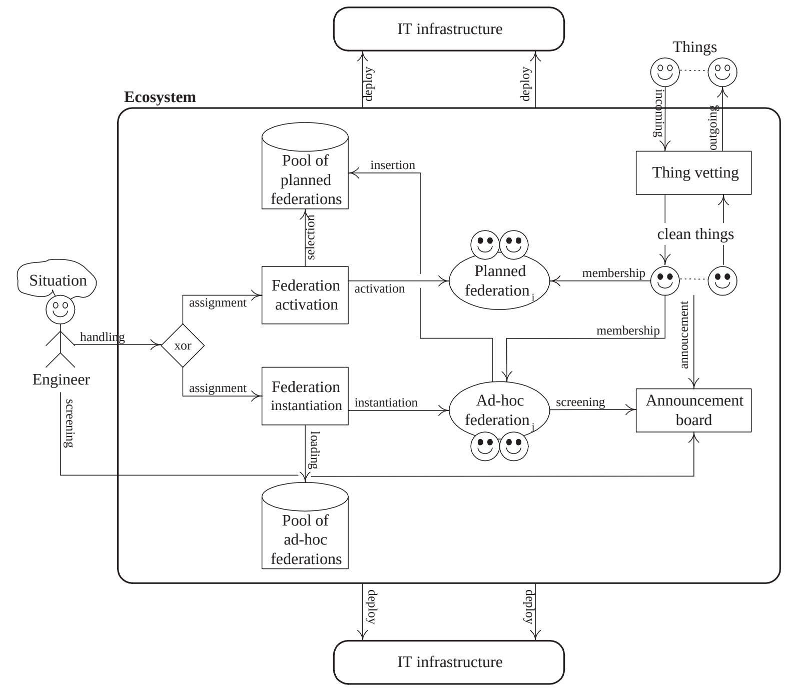


Fig. 4 Representation of the ecosystem of both things and federations of things

- Assembling stage is about identifying the necessary things that will populate federations. A thing is either concrete in a planned federation or abstract in an ad-hoc federation. During assembling, a thing's duties in a federation are enabled along with agreeing upon the collaboration means between the things in this federation.
- Storage stage is about grouping planned federations in a dedicated pool in preparation for their selection and then activation, and, also, grouping ad-hoc federations in another dedicated pool in preparation for their loading and then, instantiation. At any time, a situation is related to many ready-to-trigger planned federations and one ready-to-instantiate ad-hoc federation.
- Activation stage is about executing a planned federation following its selection from the pool of planned federations. This selection depends on

a situation's requirements. The planned federation's concrete things are already identified and now need to be put-into-action (Table 1).
xor
Instantiation stage is about searching for concrete things that will instantiate the abstract things associated with the ad-hoc federation. The search and selection of concrete things depends on the situation's requirements and are QoT-driven (Table 1).

- Disassembling stage is about putting an end to a planned federation after reviewing its performance like limited competitiveness compared to other planned federations that are all, including the one to dissemble, associated with the same situation. We recall that upon instantiating an ad-hoc federation, this one becomes a planned federation and is inserted into the pool of planned federations. Thus, disassembling targets planned federations, only. Ad-hoc federations are not subject to any dissembling.

The modules that support the functioning of the ecosystem of both things and federations of things are listed below (Fig. 4):

1. Thing-vetting module has a dual role. For the incoming flow, the vetting ensures that concrete things, first, comply with the ecosystem's regulations in order to maintain a safe and competitive ecosystem and, second, are properly described so they are assigned to federations. For the outgoing flow, the vetting ensures that concrete things do not for instance, carry any private detail or "abuse" any peer prior to leaving the ecosystem. Thanks to the vetting module, all things in the ecosystem are declared "clean". It is stated that "*The IoT era not only brings new opportunities, but also presents an expanded attach surface, already being exploited by cyber criminals*"[3] and "*IoT devices can and do get hacked regularly, and the consequences are severe*"[4].
2. Federation-activation module targets planned federations whose necessary concrete things and their duties are identified ahead of time. The ecosystem engineer identifies these things after screening the announcement board upon which concrete things post their duties. The federation-activation module is coupled to the pool of planned federations.
3. Federation-instantiation module targets ad-hoc federations whose necessary abstract things and their duties are identified waiting to be instantiated after screening the announcement board upon which concrete things post their duties. The federation-initiation module is also coupled to the pool of ad-hoc federations.
4. Announcement-board module acts as a "broker" between federations having situations' needs/requirements to satisfy/meet and concrete things having duties to offer.

Back to Fig. 1 about the 4 duties of a thing, they can be put together in a way that permits to develop composite duties as per some of the following illustrative cases:

[3] `go.armis.com/iot-security-buyers-guidev5.`
[4] `tinyurl.com/y9v5lgfq.`

1. Sensing $\rightarrow$ actuating $\rightarrow$ communicating: the outcomes of sensing are passed on to actuating whose outcomes are passed on to communicating.
2. Sensing $\rightarrow$ actuating: the outcomes of sensing are passed on to actuating whose outcomes are finals.
3. Sensing $\rightarrow$ reasoning $\rightarrow$ actuating $\rightarrow$ communicating: the outcomes of sensing are passed on to reasoning whose outcomes are passed on to actuating whose outcomes are passed on to communicating.

Composite duties illustrate how a duty's QoT properties could impact the QoT properties of other duties. For instance, a high-quality actuated outcome should lead to a better acceptance of this outcome when communicated to potential recipients. The opposite would happen when the actuation is of low quality.

3.3 Reuse and comparison

In this section, we discuss 2 cases that could help sustain the benefits of federations in the context of an IoT ecosystem. These cases are about federation reuse and federation comparison. The former helps adapt existing (planned) federations to new situations and the latter helps compare existing (planned) federations.

Federation reuse: The current definition of federation assumes a perfect match between a situation's requirements and the different things that populate the federation that handles this situation. We refer to this federation as perfectly-populated. However, it happens that an existing federation could be assigned to a new situation although this federation has either more or less things than what this situation handling requires. An existing federation that has a proven-track record of good performance, for example, could be potentially reused to handle new situations rather than developing new (ad-hoc then planned) federations from scratch. In this context, 2 options could arise:

1. Over-populated: some existing things in a federation are deemed unnecessary for a situation (based on their duties). 2 exclusive options are offered to the federation: (i) expel the unnecessary things with the risk of looking for other things (could be the expelled ones) when new situations emerge and this federation along with its initial set of things would have been an ideal candidate or (ii) compensate the unnecessary things for remaining idle in the federation, i.e., doing nothing.
2. Under-populated: all existing things in a federation are not sufficient for a situation. This triggers the search for new things at the risk of changing the internal structure (in terms of things) of the federation with respect to the situation to which this federation was initially assigned.

Federation comparison: Since several (planned) federations are associated with a situation (regardless of its non-functional requirements), it would be interesting to compare them using specific metrics that could be monitored

over an observation window (w). These metrics could address questions like what is the mostly/least included concrete thing in federations, what is the mostly/least reliable concrete thing in a federation, and what is the mostly/least requested federation of things? Another benefit of these metrics is that once an ad-hoc federation is formed and then instantiated, it could be compared to existing planned federations.
In the following, S denotes situation, F denotes federation, T denotes thing, and N denotes the total number of federations linked to a situation. N changes over time once an ad-hoc federation becomes labeled as planned or a planned federation is dismantled.

1. In-Demand metric identifies the federation that has been assigned most of (or least) the situation handling requests. This metric is measured by the number of times a federation is assigned a situation over the total number of times this situation has arisen (Equation 1).

$$\mathsf{inDemand}^{w}_{(S_i,F_j)} = \frac{|\mathsf{assign}(S_i, F_j)|}{|S_i|} \tag{1}$$

2. Satisfaction metric represents the subjective opinion of how a federation performed when assigned a situation. Persons responsible for monitoring situation completion provide opinions. This metric is measured based on the aggregate positive and negative votes that a federation has received from all persons (Equation 2).

$$\mathsf{satisfaction}^{w}_{(S_i,F_j)} = \frac{\mathsf{pos}_{(S_i,F_j)}}{\mathsf{pos}_{(S_i,F_j)} + \mathsf{neg}_{(S_i,F_j)}} \tag{2}$$

3. Activity metric represents the participation levels of a thing in the different federations to which it belongs so, that, it remains in these federations for longer periods of time. This level could be linked for instance, to a threshold that indicates when a thing should sign off from a federation. A reason could be the limited number of situation handling requests that are assigned to a federation. We formulate the activity metric based on the number of times a situation has been assigned to a particular federation in which a particular thing resides over the number of times this situation has been assigned to all federations associated with this situation and in which this particular thing resides too (Equation 3). The threshold's value varies from one thing to another.

$$\mathsf{activity}^{w}_{(S_i,F_j,t_k)} = \frac{|\mathsf{assign}(S_i, F_j)_{t_k}|}{\sum_{j=1}^{N}(|\mathsf{assign}(S_i, F_j)_{t_k}|)} \tag{3}$$

4 Thing assignment to federations

Assigning things to federations depends on things' QoT properties and federations' requirements. In this section we discuss the assignment in terms of design and implementation.

4.1 Assignment design

In a federation, operational dependency exists between all things. This dependency occurs when a thing (t_i) needs a peer (t_j) to complete its duty. For instance, input (i) is passed to t_1 that will process it before sending it to t_2 for extra processing till the final output (o) is obtained. Thus, t_2 is dependent on t_1 to complete its duty. In a real-world scenario, there will be multiple things having similar duties that could fulfill the operational dependency between things. However, the concern is how to select the most suitable thing in term of compatibility from a set of duty-wise similar things. We formally describe below how the operational dependency is handled.

Notations and definitions. 2 sets are defined.

- $\mathsf{T}_f = \{t_{f_1}, t_{f_2}, t_{f_3}, \cdots, t_{f_n}\}$ is a set of duty-wise similar things, where t_{f_x} ($1 \leq x \leq n$) is the x^{th} thing and n is the total number of things. f is a similar duty that these things perform but with different levels of quality.
- $\mathsf{QoT}_f = \{qot_{f_1}, qot_{f_2}, qot_{f_3}, \cdots, qot_{f_n}\}$ is a set of QoT properties, where qot_{f_x} is the x^{th} property and n is the total number of properties.

In addition to QoT properties, we consider an extra QoT property, $\mathsf{Comp}(t_{f_i}, t_{f_j})$, that indicates the compatibility level between 2 things t_{f_i} and t_{f_j}. The higher the better showing a strong coupling between things in term of data exchange, for example.

Similarity computation. Based on some requirements related to an ideal thing t_{f_x}, we use the set of duty-wise similar candidate things, T_f. We would like to find the best thing from this set which is similar in terms of QoT properties to t_{f_x}. To this end, we suggest Equation 4

$$\mathsf{SIM}(t_{f_x}, t_{f_i}) = \frac{1}{\sum_{qot \in \mathsf{QoT}: qot_{t_{f_i}} < qot_{t_{f_x}}} \mid qot_{t_{f_x}} - qot_{t_{f_i}} \mid} \tag{4}$$

QoT is the set of common QoT properties present in both t_{f_x} and t_{f_i}, and $qot_{t_{f_x}} - qot_{t_{f_i}}$ is the difference between the QoT property of an ideal thing and a candidate thing. $qot_{t_{f_i}} < qot_{t_{f_x}}$ condition is a threshold which tells that all QoT properties of the candidate thing should be narrowed down for summation whose QoT value is less than the ideal thing's corresponding QoT value. The greater the difference the more it negatively affects the similarity. Another factor to consider here is the compatibility that tells the coupling level of 2 things when they worked with each other in the past. This factor is defined by v whose value can be $-1 \leq v \leq 1$. If v is positive it means positive compatibility; otherwise negative compatibility. Here t_{f_i} is the candidate thing and t_{f_j} is the thing which needs t_{f_i} to complete its duty (Equation 5).

$$v = \mathsf{Comp}(t_{f_i}, t_{f_j}) \tag{5}$$

$\mathsf{Score}(t_{f_i})$ denotes the overall score of the candidate thing t_{f_i} and is calculated using Equation 6.

$$\mathsf{Score}(t_{fi}) = \frac{v}{\mathsf{SIM}(t_{fi}, t_{fj})} \tag{6}$$

Algorithm 1 is our proposed thing selection approach that is based on QoT properties and operational dependency. It takes t_{fx}, T_f, and t_{fj} as inputs and produces Score as an output. In lines 1 to 5, the difference of common QoT properties of t_{fi} and t_{fx} is summed up by considering the threshold of $qot_{t_{fi}} < qot_{t_{fx}}$. Similarity computation is done through lines 6 and 8 according to the satisfying conditions. Line 11 refers to the computation of compatibility level between t_{fi} and t_{fj}. Finally, the Score assessment is done for t_{fi} (line 12) which is dependent upon compatibility calculation and similarity computation done in the previous steps.

Algorithm 1 Computing score for candidate things

```
INPUT: $t_{fx}$, $T_f$, $t_{fj}$
OUTPUT: Score
 1: for all $t_{fi} \in T_f$ do
 2:     $Sum_{DiffOf\mathsf{QoT}} \leftarrow 0$
 3:     for all $qot \in \mathsf{QoT}$ do
 4:         if ($qot_{t_{fi}} < qot_{t_{fx}}$) then
 5:             $Sum_{DiffOf\mathsf{QoT}} \leftarrow Sum_{DiffOf\mathsf{QoT}} + \mid qot_{t_{fi}} - qot_{t_{fx}} \mid$
 6:             $\mathsf{SIM}(t_{fx}, t_{fi}) \leftarrow \frac{1}{Sum_{DiffOf\mathsf{QoT}}}$
 7:         else
 8:             $\mathsf{SIM}(t_{fx}, t_{fi}) \leftarrow 1$
 9:         end if
10:     end for
11:     $v \leftarrow \mathsf{Comp}(t_{fi}, t_{fj})$
12:     $\mathsf{Score}(t_{fi}) \leftarrow \frac{\mathsf{v}}{\mathsf{SIM}(t_{fx}, t_{fi})}$
13: end for
```

4.2 Assignment implementation

For evaluation needs, we carried out several experiments based on some in-house Python programs that we developed to simulate things. The programs run on a Dell notebook with the following technical specification: Intel(R) Core(TM) i5-2540M CPU @ 2.60GHz, 4GB RAM with Windows 10 Enterprise. These programs select things using similarity related to QoT properties (like response time and throughput) and operational dependency that allows to connect things together. Each thing has a set of response time values for past usage experiences that we average out, so that each thing has a single value. We also generated compatibility values for 200 random things with each other. These values are between -1 and 1, where -1 shows minimum compatibility (i.e., low eagerness to work together) and 1 shows maximum compatibility (i.e., high eagerness to work together).

The experiments were carried out using 2 datasets: WSDream dataset[5] and an in house dataset of randomly generated compatibility values for the 200 things. Thus, we made a dataset suitable for thing selection based on QoT properties and operational dependency.

Experiments were performed in 2 phases, In phase 1, we selected 5 random target things which are dependent on others to complete their duties. For each target thing i, we narrow down 10 candidate duty wise similar things which can fulfil the operational dependency with thing i Next, we select the most optimal thing in term of operation dependency among the selected candidate things. For example, in Table 2, candidate thing 8 with score 0.96 has been selected as the most optimal thing in term of operational dependency for target thing 5 and overall time taken for calculating the score of candidate things 1, 2, 3, 4, 11, 6, 7, 8, 9 and 10 is 0.00012 seconds

Table 2 Choosing duty-wise optimal thing with number of candidate things equal to 10

Target Thing	5	9	15	22	37
Candidate Thing 1 (Score)	1 (0.52)	5(0.99)	24(0.34)	36(-0.58)	49(0.04)
Candidate Thing 2 (Score)	2 (0.64)	17(-0.33)	36(0.32)	41(-0.25)	56(-0.4)
Candidate Thing 3 (Score)	3 (0.57)	24(-0.97)	44(-0.7)	48(0.05)	64(0.99)
Candidate Thing 4 (Score)	4 (-0.81)	25(-0.48)	58(-0.62)	57(-0.37)	77(-0.49)
Candidate Thing 5 (Score)	11(0.28)	67(-0.96)	67(-0.96)	69(0.58)	89(0.19)
Candidate Thing 6 (Score)	6 (-0.38)	75(0.72)	75(0.72)	75(0.56)	92(-0.06)
Candidate Thing 7 (Score)	7 (-0.36)	84(0.06)	82(0.38)	82(0.38)	106(0.32)
Candidate Thing 8 (Score)	8 (0.96)	69(0.51)	96(-0.01)	91(1.0)	117(-0.52)
Candidate Thing 9 (Score)	9 (-0.1)	95(-1.0)	104(-0.46)	105(-0.3)	128(0.4)
Candidate Thing 10 (Score)	10 (0-0.53)	101(-0.59)	115(-0.69)	115(-0.69)	135(-0.11)
Optimal Thing	8	5	75	91	64
Time(seconds)	0.00012	0.00022	0.00020	0.00010	0.00023

In phase 2 we selected 5 random target things which are dependent on others to complete their duties. For each target thing ii, we narrow down 15 candidate duty-wise similar things which can fulfil the operational dependency of thing i. Next, we select the most optimal thing in term of operation dependency among the selected candidate things. For example, in Table 3, candidate thing 35 with score 0.91 has been selected as the most optimal thing in term of operational dependency for target thing 6 overall time taken for calculating the score of candidate things 7, 9, 11, 13, 15, 17, 19, 21, 23, 5, 31, 35, 38, 40 and 42 is 0.0001692 seconds

[5] `wsdream.github.io.`

Table 3 Choosing duty-wise optimal thing with number of candidate things equal to 15

Target Thing	6	8	10	13	19
Candidate Thing 1 (Score)	7(0.62)	11(0.71)	12(-0.02)	15(0.99)	21(0.22)
Candidate Thing 2 (Score)	9(-0.05)	13(0.9)	14(-0.53)	17(-0.2)	23(-0.34)
Candidate Thing 3 (Score)	11(-0.1)	15(0.73)	16(0.53)	19(0.77)	25(0.37)
Candidate Thing 4 (Score)	13(0.25)	17(-0.63)	18(0.88)	21(0.41)	27(0.25)
Candidate Thing 5 (Score)	15(0.87)	19(-0.59)	20(0.64)	23(0.48)	29(0.38)
Candidate Thing 6 (Score)	17(0.89)	21(-0.68)	22(-0.41)	25(0.25)	31(0.82)
Candidate Thing 7 (Score)	19(0.78)	23(0.33)	24(0.15)	27(-0.44)	33(0.17)
Candidate Thing 8 (Score)	21(-0.39)	25(0.68)	26(-0.32)	29(0.25)	35(-0.95)
Candidate Thing 9 (Score)	23(-0.73)	27(0.44)	28(-0.58)	31(0.56)	37(0.32)
Candidate Thing 10 (Score)	5(0.13)	29(0.22)	30(0.02)	33(0.52)	41(0.63)
Candidate Thing 11 (Score)	31(-0.45)	31(0.83)	32(0.3)	35(-0.4)	43(0.07)
Candidate Thing 12 (Score)	35(0.91)	33(0.68)	34(0.57)	37(-0.21)	45(0.32)
Candidate Thing 13 (Score)	38(-0.85)	35(0.44)	36(0.54)	39(-0.92)	47(0.24)
Candidate Thing 14 (Score)	40(0.26)	37(-0.22)	38(0.07)	41(0.66)	49(-0.44)
Candidate Thing 15 (Score)	42(0.36)	39(0.83)	40(-0.14)	43(-0.1)	39(0.32)
Optimal Thing	35	13	18	15	31
Time(seconds)	0.0001692	0.0001682	0.000172	0.0002585	0.0001718

Fig 5 and Fig 6 show the time consumed for calculating score for each target thing with different number of candidate things. The x-axis for each graph represents the ids of target things and y-axis for each graph represents time in seconds

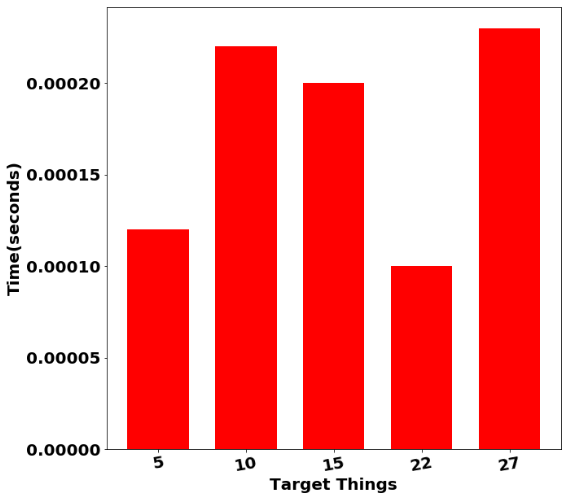


Fig. 5 Performance evaluation based on time and for 10 candidate things

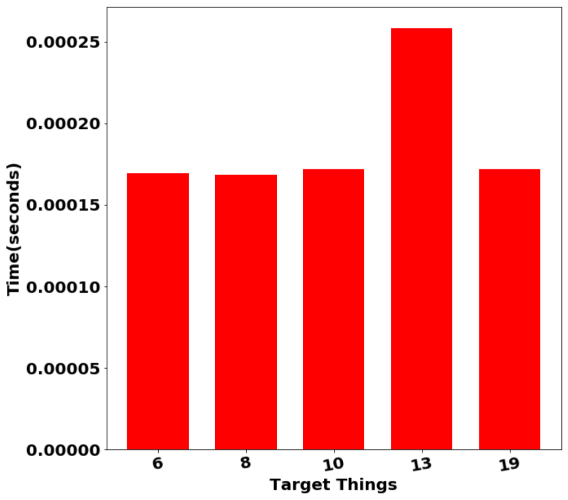


Fig. 6 Performance evaluation based on time and for 15 candidate things

The results shows that our approach takes maximum of 0.00022 seconds when the number of candidate things is 10 and 0.00025 seconds for 15 things.

5 Related work

Despite the growing interest in IoT [15], there are not, to the best of our knowledge, dedicated works that examine thing federation (except of [12]) nor thing assignment to federations using non-functional properties. The below related-work paragraphs discuss the concept of federation in different domains for instance, device, cloud, and identity.

Heil et al. define IoT as a context-aware federation of devices[6] [6]. The objective of setting-up this federation is to support users access, connect, and locate arbitrary devices according to their functionalities. Heil et al.'s thing federation is different from ours; we advocate for gathering devices/things together in response to specific situations' needs, and, not, for accessing these devices/things, only. Heil et al.'s approach takes advantage of the concept of Federated Devices Assemblies (FDX) to integrate real-world devices into service federations. This integration encapsulates and exposes devices' capabilities for external use in terms of operations, status variables, and events. According to the authors, FDXs are already designed to communicate among each other irrespectively of the hardware addressed underneath.

Mathlouthi and Ben Saoud discuss cloud federation so, that, a flexible composition of System of Systems (SoS) is enabled [11]. A SoS is about the cooperation of several constituents that are complex, heterogenous, autonomous, and independently governed, but capable at the same time of working in a

[6] "*Devices can be as small as lightbulb or as large as an airplane*" [13].

cooperative way to achieve common goals. These constituents' characteristics raise concerns with respect to interoperability, fault tolerance, continuous monitoring, etc. Because SoSs are deployed over different clouds, the federation of clouds at the software level (SaaS where the 1^{st} S could be SoS) is deemed necessary. In line with Mathlouthi and Ben Saoud, we later show that thing federation could benefit from cloud federation in the sense that thing federations could be deployed over multiple collaborative clouds when complex situations need to be handled.

*aaS where everything is software, platform, infrastructure, data, or thing federation[7] is a model that exposes "resources" to the external world through services for different reasons thoroughly discussed in the literature [17]. To allow exposing thing federation as a service, Celesti et al. discuss IoT-as-a-Service (IoTaaS) in conjunction with the development that cloud computing is going through and that is leading to IoT cloud and cloud federation [1]. The authors suggest 3 stages towards a true IoT cloud federation. The first stage, "monolithic IoT clouds", is the current stage where IoT clouds are independent; IoT devices interact with a remote cloud system that collects the sensed and actuated data coming from heterogenous IoT devices. The second stage, "vertical supply chain", requires a smart, improved coordination system to enable the cooperation of different involved IoT cloud providers. Finally, the third stage, "IoT cloud federation", calls for a logical layer between the physical infrastructure and services. 2 types of clouds exist in a federation: home IoT and foreign IoT. The former is a provider that needs extra external sensing and actuating duties and, consequently, forwards federation requests to the latter with the purpose of elastically enlarging its IoT infrastructure. Before concluding this section, it is worth noting that an IoT cloud provider could simultaneously be home cloud and foreign cloud. Finally, Celesti et al. recommend a 3 layer cloud federation reference-architecture that would meet automation and scalability, interoperable resource provisioning, and interoperable security requirements. The 3-layers are virtualization, virtual infrastructure manager, and cloud manager. The latter is capable of providing IoTaaS in the form of IaaS, PaaS, and SaaS.

Farris et al. achieve the dynamic cooperation of IoT cloud providers by proposing a model in which Mobile IoT Federations are exposed as Services (MIFaaS) [5]. The authors note that today's ICT landscape is characterized by a large number of heterogeneous devices and objects that offer a wide range of services such as computation, storage/caching, and sensing/actuating. MIFaaS is about federating private and public mobile IoT clouds and forming coalitions in these federations from an opportunistic perspective so, that, IoT cloud providers would not think of leaving the coalition. MIFaaS, also, taps into the benefits of edge computing so, that, latency during data transmission to/from remote data centers remains under control.

Torroglosa-Garcia and Skarmeta-Gomez discuss the interoperability of identity federation systems [20]. These federations unify and simplify user and

[7] With the first three defining the essence of cloud computing.

service management using trust relationships. However, the large number of federations, each focussing on different areas, necessitates their interoperability to ensure a consistent access across all these federations, and, hence, the same digital identity is used. Identity federation systems are like our proposed collaboration of planned and ad-hoc federations.

6 Conclusion

This paper presented an approach for assigning things that will populate federations in charge of handling real-life situations like tunnel closure. The assignment relied on a set of non-functional properties that define a thing's QoT model. These properties describe the performance of a thing's potential duties namely sensing, reasoning, actuating, and communicating. In response to situations' requirements, 2 types of federations were proposed, which are planned and ad-hoc. Federations transition through 4 stages, assembling, storage, **either** activation **or** instantiation, and ultimately disassembling, in which specific actions are executed like checking the credentials of things and forming federations. A set of experiments using a mix of real and simulated datasets allowed to demonstrate the technical doability of the thing assignment approach. In term of future work we would like to extend the QoT model to federations so, that, similar federations could be compared and then assigned to situations.

References

1. A. Celesti, M. Fazio, M. Giacobbe, A. Puliafito, and M. Villari. Characterizing Cloud Federation in IoT. In *Proceedings of the 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA'2016)*, Crans-Montana, Switzerland, 2016.
2. P. Charith, L. Chi Harold, J. Srimal, and C. Min. A Survey on Internet of Things From Industrial Market Perspective. *IEEE Access*, 2, 2014.
3. M. Daniel A. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6), 2002.
4. M. Eisa, M. Younas, and K. Basu. Analysis and Representation of QoS Attributes in Cloud Service Selection. In *Proceedings of the 32nd International Conference on Advanced Information Networking and Applications (AINA'2018)*, Cracow, Poland, 2018.
5. I. Farris, L. Militano, M. Nitti, L. Atzori, and A. Iera. MIFaaS: A Mobile-IoT-Federation-as-a-Service Model for Dynamic Cooperation of IoT Cloud Providers. *Future Generation Computer Systems*, 70, 2017.
6. A. Heil, M. Knoll, and T. Weis. The Internet of Things - Context-based Device Federations. In *Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS'2007)*, Hawaii, USA, 2007.
7. K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, and N. Mavridis. Microservice-Based IoT for Smart Buildings. In *Proceedings of AINA 2017 Workshops held in conjunction with the 31st International Conference on Advanced Information Networking and Applications (AINA'2017)*, Taipei, Taiwan, 2017.
8. L. Lin, P. Li, X. Liao, H. Jin, and Y. Zhang. Echo: An Edge-Centric Code Offloading System With Quality of Service Guarantee. *IEEE Access*, 7, 2019.
9. Z. Maamar, T. Baker, M. Sellami, M. Asim, E. Ugljanin, and N. Faci. Cloud *versus* Edge: Who Serves the Internet-of-Things Better? *Internet Technology Letters, Wiley*, June 2018 (https://tinyurl.com/y767ybor).

10. Z. Maamar, K. Boukadi, E. Ugljanin, T. Baker, M. Asim, M. Al-Khafajiy, D. Benslimane, and H. El Alaoui El Abdallaoui. Thing Federation as a Service: Foundations and Demonstration. In *Proceedings of the 8th International Conference on Model and Data Engineering (MEDI'2018)*, Marrakesh, Morocco, 2018.
11. W. Mathlouthi and N.B. Ben Saoud. Flexible Composition of System of Systems on Cloud Federation. In *Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud'2017)*, Prague, Czech Republic, 2017.
12. O. Mroβ and K. Meiβner. Towards Distribution Options in the End-User Development of Multi-device Mashups. In *Proceedings of the 2nd International Workshop on Engineering the Web of Things (ENWOT'2018) held in conjunction with the 18th International Conference on Web Engineering (ICWE'2018)*, Cáceres, Spain, 2018.
13. Editor's Note. The Age of the Internet of Things. *Computing edge*, 4(10), October 2018.
14. M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer*, 40(11):38–45, November 2007.
15. C. Perera, C.H. Liu, S. Jayawardena, and M. Chen. A Survey on Internet of Things From Industrial Market Perspective. *IEEE Access*, 2, 2014.
16. M.A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke. Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, 3(1), 2016.
17. J.W. Rittinghouse and J.F. Ransome. *Cloud Computing: Implementation, Management, and Security.* Taylor & Francis, 2009.
18. J. Seeger, R.A. Deshmukh, and A. Bröring. Running Distributed and Dynamic IoT Choreographies. *CoRR*, abs/1802.03159, 2018.
19. A.S. Thuluva, A. Bröring, G.P. Medagoda, E. Don, D. Anicic, and J. Seeger. Recipes for IoT Applications. In *Proceedings of the 7th International Conference on the Internet of Things (IoT'2017)*, Linz, Austria, 2017.
20. E.M. Torroglosa-Garcia and A.F. Skarmeta-Gomez. Towards Interoperabilty in Identity Federation Systems. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 8(2), June 2017.