Online ISSN 1848-3380, Print ISSN 0005-1144
ATKAFF 57(2), 304–318(2016)

Krešimir Mišura, Mario Žagar

Negotiation in Internet of Things

DOI 10.7305/automatika.2016.10.1193
UDK 004.775:331.106.42

Original scientific paper

Internet of things as a market, and the number of connected devices in particular is growing very rapidly. Currently, application owners deploy new devices for each application that needs the data. As the number of sensors increases, it will become much more practical to reuse existing sensors for new applications than to deploy new ones. But the problem is that the application owner needs to agree with device owners on conditions under which will the data be made available to applications. Doing this manually is very expensive both in terms of money and time. We implemented a system that does this automatically using negotiating agents. The system was tested on simulated environments and showed that it can mediate between devices and applications with reasonable performance.

Key words: Internet of things, Negotiation, Software Agents

Pregovaranje u Internetu stvari. Internet stvari kao tržište, a posebno broj spojenih uređaja, raste vrlo brzo. Danas vlasnici aplikacija postavljaju nove uređaje za svaku aplikaciju kojoj su potrebni podatci. Kako se povećava broj senzora u upotrebi, postaje sve praktičnije koristiti postojeće senzore nego postavljati nove. Problem predstavlja činjenica da se vlasnik aplikacije mora dogovoriti s vlasnicima uređaja o uvjetima pod kojima će aplikacijama biti dozvoljeno dohvaćanje izmjerenih vrijednosti. Pojedinačno je dogovaranje između vlasnika za svaki uređaj skupo i sporo. Izgradili smo sustav koji automatizira ovaj proces pomoću programskih agenata koji pregovaraju. Sustav je ispitan na simuliranom okruženju i pokazuje da može posredovati između uređaja i aplikacija s razumnim performansama.

Ključne riječi: Internet stvari, Pregovaranje, Programski agenti

1 INTRODUCTION

The basic idea of Internet of Things (IoT) concept is the pervasive presence around us of a variety of things or objects such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. which are able to interact with each other and cooperate with their neighbors to reach common goals [1]. More specifically, in this paper we are referring to IoT as a loosely coupled, decentralized system of smart objects - that is, autonomous physical/digital objects augmented with sensing, processing and network capabilities [2]. Web of Things is a related concept to Internet of Things. Unlike IoT which only assumes device connectivity on the IP layer, Web of Things assumes that devices are accessible using Web protocols like HTTP. In [3,4] the authors define WoT devices as first-class citizens of the Web. They consider WoT as a refinement of the IoT by integrating smart things not only into the Internet (the network), but into the Web (the application layer). Even though the proof-of-concept implementation presented in this paper uses Web connected devices, in general negotiation strategies between device and application owners can be applied just as well for devices that aren't available using Web protocols so we use the term 'Internet of Things' in the remainder of this paper.

There is already a lot of Internet connected devices, and this number is expected to grow rapidly [5]. This multitude of devices makes it very important to have a simple and efficient mechanism for applications to find devices and data they can use.

Multiple discovery architectures were already proposed in the literature [6, 7], but they assume that devices offer their data publicly by the very act of registering to the discovery services. There is no mechanism that would allow device owner to specify the amount of money per measurement for which he would be willing to share sensor readings.

When discovering appropriate devices, users requirements generally fall in two distinct categories, non-negotiable and negotiable [8]. For example, if a user is interested in measuring the temperature at certain coordinates, he will certainly not be satisfied with a humidity or pressure measurements. This is an example of a non-

negotiable requirement. However, he is probably willing to accept measurement from a sensor that is nearby even though not at the exact coordinates, if the price is lower or some other conditions are more suitable. This paper deals primarily with negotiable requirements.

We propose a marketplace, where applications buy data from devices, and autonomous agents are used to find the exact conditions (price, time, etc.) under which the data is provided. They simply need to agree on the price (and possibly other criteria such as time of delivery). We named this marketplace and supporting systems *'IoT Mediator Platform'*. Virtual market and negotiation problems were already investigated in detail by the MultiAgent community [9–12]. We believe that it is possible to use the same mechanisms in the IoT domain to create dynamic contracts between devices and applications, as shown in Figure 1.

One interesting use-case of this marketplace is creating a city heart rate monitor where marketing agencies could gather real time heart rate data in a city and use that data in their marketing campaigns for example to show how the city heart rate rises during important sporting events or other situations of public interest. But, for that to work, the application needs a lot of devices to provide it data. Deploying new devices for this specific purpose is not financially viable, and personally contacting many device owners to find the ones interested in exposing the data and then negotiating with them on the price, schedule and other conditions is tiresome and unscalable. Creating a mediation framework that connects the compatible devices and applications and performs the negotiation automatically would ease this process significantly, and could create many new economically viable use cases for the IoT.

The aim of this paper is to show that using negotiating agents, data provision terms can be found that both application and device owners find acceptable.

We already introduced the idea of negotiation between IoT devices and applications to find acceptable terms of data provision in a previous paper [13]. The original scientific contributions of this paper over previous are: 1) detailed analysis of algorithms that are used in agent negotiation 2) analysis of how credibility can be utilized in IoT context to defend against malicious and malfunctioning devices and 3) use of TwoTariffs algorithm to reduce energy usage and increase earnings.

The remainder of this paper is organized as follows. A motivational example for building this platform is presented in Section 2. A short description of software agents and JADE middleware platform is given in Section 3. Section 4 examines the previous research done on this and similar subjects. Platform architecture is presented in Section 5. Section 6 describes the actual negotiation in more detail. Various negotiation algorithms are described in Section 7. Section 8 gives implementation details. Results

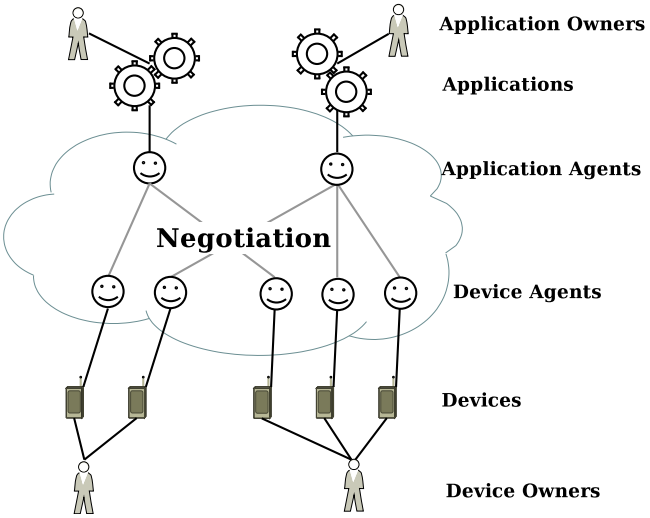

Fig. 1. Platform overview

are presented in Section 9. Directions for future work and concluding remarks are given in Section 10.

2 MOTIVATION

Although there are many use-cases for the platform presented in this paper, in this section we present one that would help clarify the motivation behind this project. The idea was to create an Internet connected device that would be making environmental measurements for the owner. It measures dust concentration, temperature, humidity, brightness levels, and has carbon monoxide and flammable gasses detection. Users would buy this device and install it on their balcony or garden. This would provide them with useful atmospheric measurements just outside their home.

However, they can go one step further and try to make money with that measurements. They would register the device to the IoT Mediator Platform and specify the conditions under which they are willing to share data with third-party applications. Applications can dynamically search for, and use the data provided by this and similar devices. The IoT Mediator Platform takes care of negotiating between applications and devices, and tries to find conditions with which both of them will be satisfied. This way, applications can reuse the crowdsourced data at a significantly lower price than if they bought and deployed all devices themselves.

Dust (i.e. particle count) and Carbon Monoxide measurements could be useful to people suffering from asthma. Research shows, that fine particle concentration can increase the likelihood of asthma relapse [14]. Actually, urban pollution proves as such an important problem that European Commission in Directive 96/62/EC established

the basic principles of a common strategy to define and set objectives for ambient air quality [15]. Air pollution is usually monitored by highly reliable networks of fixed stations. However, permanent monitoring stations have a high associated acquisition and maintenance costs, which limits the number of such facilities, resulting in non-scalability of the system and in an extremely limited spatial resolution of the pollution maps. By crowdsourcing data from already deployed devices, measurements could be made with much higher resolutions and lower costs. It is easy to envision a web application that would collect information from these sensors, and provide valuable information, such as current risk for people with asthma, particle concentrations in different parts of town, prediction of concentrations in different times of the day, etc.

Temperature, humidity and brightness level measurements can be used to create a more detailed weather forecasts. Meteorological agencies would use this large amount of crowdsourced data, and incorporate it into their weather prediction models, making them finer grained and ultimately more precise, while still much cheaper than if they deployed additional weather stations.

3 SOFTWARE AGENTS

An agent is a special software component that has autonomy that provides an interoperable interface to an arbitrary system, and/or behaves like a human agent, working for some clients in pursuit of its own agenda. Usually systems consist of multiple interacting agents called multi-agent systems (MAS). Agents can decide to cooperate for mutual benefit or may compete to serve their own interests. Therefore, an agent is *autonomous*, because it operates without the direct intervention of humans or others and has control over its actions and internal state. An agent is *social*, because it cooperates with humans or other agents in order to achieve its tasks. An agent is *reactive*, because it perceives its environment and responds in a timely fashion to changes that occur in the environment. And an agent is *proactive*, because it does not simply act in response to its environment but is able to exhibit goal-directed behavior by taking initiative.

Agent-Oriented Programming (AOP) is a software paradigm that brings concepts from the theories of artificial intelligence into the mainstream realm of distributed systems. AOP essentially models an application as a collection of components called agents [16]. AOP is used in IoT Mediator Platform to accomplish agent negotiation. Rather than developing core agent communication infrastructure, we relied on agent-oriented middleware named JADE (Java Agent DEvelopment framework).

JADE is the middleware developed by TILAB for the development of distributed multi-agent applications based on the peer-to-peer communication architecture. The environment can evolve dynamically with agents, that appear and disappear in the system according to the needs and the requirements of the application environment. Communication between the agents is completely symmetric with each agent being able to play both the initiator and the responder role. JADE is developed in Java. Each instance of the JADE run-time is called a *container*. The set of all containers is called a *platform* and provides a homogeneous layer that hides from agents the complexity of the underlying tiers (hardware, operating systems, types of network, JVM). JADE allows each agent to dynamically discover other agents and to communicate with them according to the peer-to-peer paradigm. Agents communicate by exchanging asynchronous messages. The structure of a message complies with the ACL (Agent Communication Language) language defined by FIPA [17] and includes fields needed to support complex interactions and multiple parallel conversations [18].

4 RELATED WORK

There are currently many deployed IoT platforms. Most of them provide the ability to connect devices using web protocols, collect and store readings, generate alarms and actions based on the data received, and create some kind of data analysis such as aggregating the data and displaying it to users represented as graphs, pie charts etc. Examples of such platforms are Axeda[1], Carriots[2], Evrythng[3] and others.

Additionally, platforms like DeviceCloud (Etherios)[4], One platform (Exosite)[5], and Xively[6] provide device management solutions. This way, device configuration can be changed without the need for physical access to the device itself.

Sensing as a Service is a model that was modeled by other *Everything as a Service* (XaaS) services like *Infrastructure as a Service*, *Platform as a Service* and *Software as a Service* [19, 20]. XaaS is growing in popularity because of the cost effectiveness that it gives clients, where clients pay only for resources they use. Other benefits include business agility, elasticity, reliability and less maintenance work. Clients can thus focus more on their core competencies instead of dealing with lower level functionalities. Proponents of the Sensing as a Service model argue that the same reasoning will lead to using IoT devices as a service. The negotiation platform presented in this paper works very good as method of providing incentive

[1] http://www.axeda.com/
[2] https://www.carriots.com/
[3] http://www.evrythng.com/
[4] http://www.etherios.com/products/devicecloud/
[5] http://exosite.com/products/onep
[6] https://xively.com/

for device owners to utilize their sensors in Sensing as a Service applications. By using negotiating agents we provide a method to find conditions (both financial and other) that both device and application owners find acceptable and thus provide incentive for device owners to provide data their devices generate. Without IoT Mediator Platform, device owners were left with three basic options: keep the data private only for their application, make the data public for everyone to access or manually select the consumers that are allowed to access the data. With IoT Mediator Platform, they are able to automate this process with an Autonomous agent, increasing the attractiveness of Sensing as a Service concept.

In [7] the authors present DiscoWoT, an extensible discovery mechanism that incorporates multiple discovery strategies to semantically map Web resources and allows users to extend the pool of available strategies at runtime. As the authors point out, the task of finding relevant smart things is significantly more complicated than searching for documents because of several reasons: First, smart things should be identified according to dynamic, contextual information. Second, smart thing does not necessarily express its functionality such that it may be found by traditional search engines. Third, we require a mechanism that allows machines to discover devices and understand their capabilities in order to enable automatic usage by software applications. DiscoWoT is a web service that accomplishes the task of device discovery by allowing smart devices to provide semantic descriptions of the services they offer. One of the most important features of DiscoWoT is the extensible architecture that allows the injection of new discovery strategies during runtime of the system. This gives the service enough flexibility to handle any future semantic markup languages. However, as good as DiscoWoT is at discovery, once devices are discovered, it provides no way of negotiating on the conditions under which the data will be shared. Thus, it is complementary with the IoT Mediator Platform, and integrating the two could prove beneficial, and provide an interesting scientific endeavor.

A system that uses negotiation between autonomous agents to facilitate inter device communication is described in [21]. The basic scenario is of an M2M environment containing intelligent server and a user device. The intelligent server manages information of services it offers to users. It has JADE-based agents which communicate with the user device through ACL messages. Once the user device establishes a physical connection, the intelligent server conducts the processing of received requests and provides the user device with an optimized service. While this work uses a lot of the same technologies as our work, there are some significant differences. The purpose of [21] is to create a service framework that can offer optimized services to users. The proposed system uses ontologies that are used to decide which service offered by the intelligent server is optimal for the user and to get the information of that service. Agent negotiation is used to choose the optimal service from a set of predefined services, i.e. there is only one service provider. In our work, negotiation is used in a different way, because the service consumer negotiates with many different service providers to find the one that is willing to make the best offer. Also, [21] describes a local M2M environment with one service provider and possibly multiple service consumers. In our work, there is a central negotiation platform that all of the devices and application register to. Lastly, the use case described in [21] revolves around providing the optimal service to user's mobile device, while our work focuses on a general negotiation platform that can be used by any type of application that needs data that registered devices provide.

In [10], a formal model of negotiation between autonomous agents is presented. The model defines a range of strategies and tactics that agents can employ to generate initial offers, evaluate proposals and offer counter proposals. The paper discusses service-oriented negotiation. In this context, one agent (the client) requires a service to be performed on its behalf by some other agent (the server). Negotiation involves determining a contract under certain terms and conditions. Authors also present rich and flexible negotiation schemes. The models are based on realistic assumptions for autonomous computational agents and are empirically evaluated. The models and theoretical grounding presented are very useful in our IoT negotiation implementation. Most of the ideas presented in [10] are directly translatable to our domain, which is to be expected since IoT device-application negotiation is just a special case of a more general autonomous agent negotiation. Namely, algorithms we present in Section 7 are instances of Imitative Tactics named Relative Tit-for-Tat analysed in [10], and TwoTariffs is a modified Resource-Dependent tactic where the limited resource is remaining battery power on the device.

Sensor Web Enablement (SWE) is an Open Geospatial Consortium (OGC) initiative framework of open standards for exploiting Web-connected sensors and sensor systems [22]. SWE develops standards for discovery, exchange, and processing of sensor observations, and tasking of sensor systems. It includes the following standards: Observations & Measurements Schema (O&M), Sensor Model Language (SensorML), Transducer Markup Language (TransducerML), Sensor Observations Service (SOS), Sensor Planning Service (SPS), Sensor Alert Service (SAS) and Web Notification Service (WNS).

OneM2M attempts to standardize a common M2M horizontal service layer across a number of industry verticals of globally applicable M2M services [23]. The problem it is trying to solve is of each M2M solution using a propri-

etary system comprising all layers, from physical to application and developing the same services again instead of reusing existing services.

The SSN ontology is an OWL 2 ontology to describe sensors and observations developed by the W3C Semantic Sensor Network Incubator group (SSN-XG) [24]. It describes the capabilities and properties of sensors, the act of sensing and the resulting observations. It is compatible with OGC standards and can be used to enable the Semantic Sensor Web [25].

Agent negotiation was already used for determining Quality of Service guarantees in context of IoT Cloud Services [26]. Authors analyse two existing negotiation strategies, *Concession* and *Tradeoff*. Concession strategy is when an Agent relaxes some of his requirements in order to make his offer more acceptable to other parties. Tradeoff strategy is when an agent decreases his demands for some less important property but demands more on another property that is more important to him. Concession strategy has higher success rate and tradeoff strategy has higher achieved utility. Authors propose a mixed strategy where agent mixes the two basic strategies presented previously. Paper concludes that when a party has no knowledge of the strategy of its counterpart, a mixed strategy outperforms a concession one in terms of utility and a tradeoff one in terms of success rate.

Negotiation mechanisms have also been used for enabling self-organization in IoT [27, 28]. In [27], authors define a networking approach called 'goal-driven networking' to increase the network performance and simplify the configuration of networks. In [28], authors present a reference framework for management of large-scale IoT deployments. The framework promotes management that mediates between application requirements and physical infrastructure using a management as a service platform. However, such negotiation implementation is different from price-based negotiations as used in our paper, so it can't be readily applied to our case.

5 ARCHITECTURE

The IoT Mediator Platform consists of three modules: HTTP interface, database and the negotiation module. This can be seen in Figure 2. HTTP module allows devices, applications and humans to communicate with the platform using a REST interface. It is possible to register the device or application, fetch the contracts that the device or application has agreed to, and of course to unregister the device/application.

Devices register by making a POST request to the IoT Mediator Platform. HTTP message body contains device description in XML format. An example of the device description is given in Figure 3. Platform parses that message and saves device details to a database. System responds with a unique id that was associated with the newly registered device. The device can use this id for identification in all future interactions. A described XML is used for simplicity, it can be integrated with existing standards described in Section 4, but that would be a separate project that would draw away from the aim of this paper which focuses on the negotiation mechanism. Thus we leave the integration with existing standards and device description formats for future work. Tags in the XML have the following meaning:

- <type> is an enumeration indicating the type of measurement such as "temperature", "humidity", "wind speed" and similar.
- <location> specifies coordinates in standard decimal representation and elevation is given in meters
- <algorithm> specifies the algorithm that agent uses for negotiation and contains all of the relevant parameters used for algorithm configuration

Device can get all of it's contracts by doing a GET query to the IoT Mediator Platform REST endpoint, providing its unique id. System queries the database to find all of the contracts that pertain to the requesting device. From these contracts, the device knows what measurements it needs to make and at what times. The devices are responsible for fetching the contracts from the platform. This is because sending a push notification to the device would be energy inefficient in the general case considering that a lot of devices spend most of the time in sleep mode. XML serialization of a contract is shown in Figure 4. The XML description contains all of the information needed for negotiation. In 4 tags are defined as follows:

- <applicationId> and <deviceId> identify the application and device that are making the contract
- <measurementType> is the type of measurement that device will provide
- <calendar> specifies the agreed schedule at which devices will provide the data. <startTime> is in milliseconds since epoch and period is also in milliseconds.
- price is in currency defined by the IoT Mediator Platform and all contracts use that currency so there is no need to specify it in the contract XML.

Applications use the same HTTP interface for registration. When registering, application description with all of the needed parameters is XML encoded in HTTP message body. This description contains all information necessary for finding the appropriate devices and negotiating

with them. This data is sent towards database and the negotiation module. An example of XML that is sent to the IoT Mediator Platform to register an application is shown in Figure 5. The explanation of some of the properties in XML is as follows:

- <calendarWithTolerance> is similar to <calendar> property discussed earlier, but in addition to ideal schedule it specifies a <tolerance> property in milliseconds. This property means that all measurements that are within <tolerance> milliseconds from the ideal schedule are also acceptable.
- <area> is similar to location with addition of tolerance properties for coordinates and elevation which function in the same way as with calendar. Thus, all devices less <coordinateTolerance> degrees and <elevationTolerance> meters away from the ideal location are acceptable.

Negotiation module is composed of two sub-modules. A preselection module queries the database and filters the devices according to preselection conditions. This way, a smaller number of devices actually enters the negotiation phase which reduces the negotiation complexity. After that, an autonomous software agent is created for the application and for each device. The actual negotiation takes place inside the JADE container. A separate container is created for each application request, i.e. each time the application contacts the IoT Mediator Platform that it needs to find some devices, a new JADE container is created. Thus, the container contains a single application agent. It also contains multiple device agents and a single notary agent. The negotiation flow is described in Section 6. After the negotiation is finished, final contracts are saved to the database and sent back to the HTTP module. HTTP module sends the contracts back as an HTTP response.

When the device has the data ready, it sends it to the IoT Mediator Platform. The platform will save the data and provide it to the applications that have the appropriate contracts. The data needs to pass through the IoT Mediator Platform (instead of going directly from the device to the application) so that it is possible to identify devices and applications that don't stick to their ends of the contracts.

Making sure contract is respected by all parties is a complicated problem. There is existing research on automated contract creation and some interesting solutions like WSLA Framework [29] have emerged. Usually, the sides agree on the penalties in the case that one of the sides doesn't deliver or provides a service of low quality. In the case of IoT Mediator Platform, this is too heavyweight because applications and devices will typically make a large amount of contracts with a multitude of partners. Agreement with each device on the penalties would further complicate the negotiation process, and the problem would still

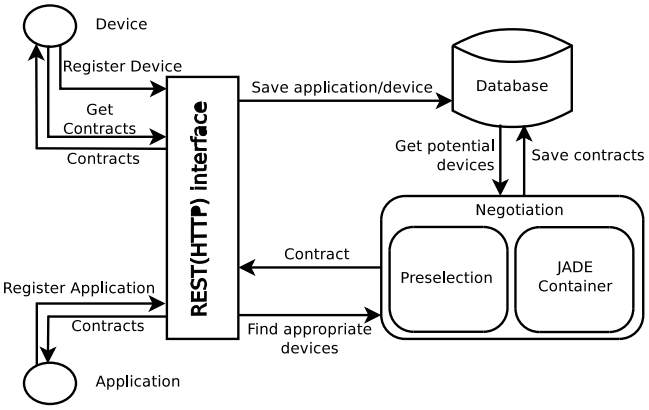

Fig. 2. Architecture overview

```
<device>
    <name>kitchen_thermometer</name>
    <type>HUMIDITY</type>
    <location>
        <latitude>45.8167</latitude>
        <longitude>15.9833</longitude>
        <elevation>120</elevation>
    </location>
    <algorithm type="minPrice">
        <minPrice>22.85</minPrice>
    </algorithm>
</device>
```

Fig. 3. Device registration XML example

remain if the agent/owner simply ignores the penalties after failing to deliver the service.

Because of that, IoT Mediator Platform implements a simpler solution based on credibility, similar to the rating system implemented on eBay and other Internet based markets. System simply notes each time the device/application fails to provide the service it has agreed upon. This is saved and available in all future negotiations. Ratings are kept on a per-owner basis. The owner can naturally have many deployed devices/applications. But if one device breaks the contract, the credibility of all other devices that belong to the same owner is degraded. How algorithms use this credibility value during negotiations is described in more details in Section 7.

6 NEGOTIATION FLOW

Negotiation is done by using the Contract Net Protocol (CNP). In the CNP, the network is assumed to consist of loosely coupled asynchronous agents, each agent can communicate with every other agent by sending messages. In the role of contractor, an agent can send requests for bids

```
<contract>
  <applicationId>12</applicationId>
  <deviceId>145</deviceId>
  <measurementSpecification>
    <id>0</id>
    <measurementType>
      TEMPERATURE
    </measurementType>
    <calendar>
      <startTime>
          1393131600000
      </startTime>
      <period>3600000</period>
    </calendar>
    <location>
      <latitude>45.8167</latitude>
      <longitude>15.9833</longitude>
      <elevation>120</elevation>
    </location>
    <price>28</price>
  </measurementSpecification>
</contract>
```

Fig. 4. Contract XML example

on each specific task to all the other agents, select the most appropriate bid and allocate the task to that subcontractor. In our implementation, application agents send requests for bids, and device agents compete on who can offer the best bid. More info about Contract Net Protocol can be found in [30].

First, user registers the application to the IoT Mediator Platform. During registration, user supplies application details via an XML document. The system parses this document and concludes on what measurements the application needs. Then, it selects all devices that potentially could provide the data the application needs. For each 'preselected' device, a device agent is created. Once this is done, negotiation process can start. The flow is depicted in Figure 6.

Application agent sends the call for proposal (CFP) to all device agents. Call for proposal contains the description of a measurement that needs to be performed. It defines the type of measurements, calendar at which the measurements need to be made and the description of acceptable geographical locations.

Each device agent analyses the CFP and creates an offer. Offer has the exact specification of measurement times, location, and price per measurement. This offer is sent back to the application agent.

The application agent can accept the offer, reject it, or create a counter-offer. This exchanging of counter-offers

```
<application>
  <name>Weather app</name>
  <measurement-requirements>
    <requirement>
      <measurementType>
        ATMOSPHERIC_PRESSURE
      </measurementType>
      <calendarWithTolerance>
        <calendar>
          <start>1393995600000</start>
          <period>3600000</period>
        </calendar>
        <tolerance>1800000</tolerance>
      </calendarWithTolerance>
      <area>
        <center>
          <latitude>42.6621</latitude>
          <longitude>114.2432</longitude>
          <elevation>120</elevation>
        </center>
        <coordinateTolerance>
          0.01
        </coordinateTolerance>
        <elevationTolerance>
          2
        </elevationTolerance>
      </area>
    </requirement>
  </measurement-requirements>
  <algorithm type="fixedBudget">
    <totalBudget>100</totalBudget>
    <minCredibility>0.95</minCredibility>
  </algorithm>
</application>
```

Fig. 5. Application registration XML example

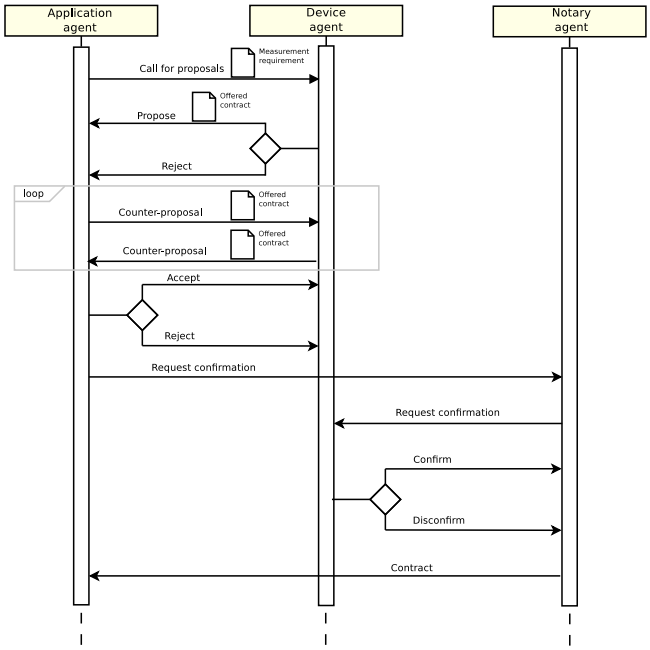

Fig. 6. Negotiation flow

can repeat until the device or application agent finally accepts or rejects the offer. Usually the agents will make concessions until they finally find some conditions that they are both satisfied with. Once the application agent finishes the negotiations with all of the agents, it sends a request to the Notary Agent asking him to validate the negotiation results and create contracts.

Notary Agent is used as a trusted third party between device and application agents. It creates contracts and guarantees that both sides agreed to them. Once notary agent receives a list of accepted offers from the application agent, it iterates through them and for each offer, requests confirmation from the concerned device agent. If the device agent confirms that the offer is indeed valid, Notary Agent creates a contract. This contract is sent back to the application agent, REST module, and database. REST module sends the contracts in a HTTP response to the user. Thus, the application knows from which devices it can get it's data.

7 NEGOTIATION ALGORITHMS

When registering an application or device, user can choose which algorithm will that application/device use for negotiating on the data usage terms. Application owners can choose between two algorithm templates: FixedBudget and ByPriority. Device owners can also choose between two template algorithms: MinPrice and TwoTariffs. In addition to specifying the algorithm, the user must provide algorithm parameters, such as budget in dollars, or minimal price etc.

Regardless of the chosen template or the specified parameters, all algorithms use a modification of a well-known tit-for-tat algorithm for creating counter proposals. They all have a starting price that was specified when the application/device was registered, or calculated from other specified parameters. If the agent is creating an initial offer, it will use this start price. The specifics of creating counter-offers are described below for application and device agents separately.

7.1 Application Algorithms

First, application agent sends request for bids to all device agents. After receiving responses, it selects the best one and continues negotiations with that device agent. Target or start price is the price at what the application would like to purchase the measurement. This is the price with what the application agent will begin the negotiations (initial offer), and actual price will usually be higher than target price because of the compromises the agent will make during negotiations. If creating a first counter-offer, application agent will increase his initial offer for 10% of the difference between his initial offer and that offered by the device agent. If there were already multiple counter-offers, the application agent will increase his last offer by the same amount as the one by which the device agent has decreased his last offer. This is best described by the formula 1. d_n is the device agent's offer in step n, while a_n is application agent's offer in step n.

$$a_n = a_{n-1} + \frac{d_{n-1} - d_n}{d_{n-1} - a_{n-1}} * (d_n - a_{n-1}) \qquad (1)$$

This process continues until device and application agent reach a common price. In order to avoid the prices closing in asymptotically, they are considered equal if they differ by less than 1 percent.

Application owners can choose between two template algorithms: FixedBudget and ByPriority.

FixedBudget is a simple algorithm where the application agent tries to satisfy as much measuring requirements as it can with the given budget (B). User defines the budget amount. User can also specify the target price. If target price is not provided by the user, Equation 2 is used to calculate it. N_{req} represents the number of measurement requirements that the agent was given.

$$a_0 = \frac{1}{2} * \frac{B}{N_{req}} \qquad (2)$$

If the user wants to satisfy all of the requirements, no matter what the price, he can leave out the budget in application specification. Representing agent will then act as if

the budget is unlimited, and try to satisfy all of the requirements. In this scenario, the user must provide a starting negotiation price for each requirement. During negotiations, representing agent first concludes cheapest deals and then moves to more expensive ones, in order to maximize the number of satisfied requirements with the given budget.

ByPriority is similar to FixedBudget, difference being in the ordering of requirements that the agent will try to satisfy. It is assumed that the user has provided requirements sorted by importance, first being the most important. Agent will respect this, and only negotiate on a requirement after it has satisfied all of the higher priority requirements. This might mean that if the price for first couple of requirements is very large, it will be able to satisfy only a couple of requirements before it depletes it's budget. Other than difference in ordering, ByPriority is exactly the same as FixedBudget.

7.2 Device Algorithms

Reserve price is the smallest price at which the device agent will sell the measurements. It is defined by the device owner during device registration. Target price is defined as 1.5 * reservePrice. If creating a first counter-offer, device agent will decrease his initial offer for 10% of the difference between his initial offer and that offered by the application agent. If there were already multiple counter-offers, the device agent will decrease his last offer by the same amount as the one by which the application agent has increased his last offer. This is best described by the formula 3. Symbols are the same as in the last example.

$$d_n = d_{n-1} - \frac{a_n - a_{n-1}}{d_{n-1} - a_{n-1}} * (d_{n-1} - a_n) \qquad (3)$$

Device owners can choose between MinPrice and TwoTariffs algorithms.

MinPrice is a very simple algorithm that accepts all offers that pay at least the specified amount of money per measurement. It doesn't care at what time the measurements need to be made. All offers with less than the specified amount are rejected. User defines the minimal amount of money needed for a measurement to take place.

TwoTariffs is an algorithm that defines a cheaper price R_c for measurements that can be made at moments when the device is already awake, and a more expensive price R_e otherwise. This makes the TwoTariffs algorithm ideal for energy savings. Remote sensors typically consume much more power when communicating than when idle [31].

Other than just communication costs, some sensors spend a great deal of energy for the actual measurement. Example of those are the electrochemical gas sensors. They measure the concentration of a target gas by oxidizing or reducing the target gas at an electrode and measuring the resulting current [32]. They have an internal heater that uses power when heating.

If we could use the same communication cycle to pass measurements to more than one application, this would significantly reduce the energy consumption. By making the measurements at already awake time cheaper than regular measurements, this algorithm makes the application agents reuse the same wake cycles for different applications.

User needs to define what are the values of R_c and R_e. The reserve price R is calculated based on the ratio (p) between the number of measurements that can be made without additional waking of the device, and total number of measurements. In the case all of the measurements can be made when the device is already awake p will be 1, and in case when each measurement requires waking the device, p will be 0. The equation used to calculate the value of reserved price is given in 4.

$$R = R_c + (R_e - R_c) * p \qquad (4)$$

Although negotiation algorithms presented above are just variations of the well known tit-for-tat algorithm [10], to our knowledge they haven't been used in negotiation on data provision in IoT context. In addition to that, TwoTariffs algorithm was specifically developed in this paper for IoT use-case and constitutes an original contribution. It reduces energy consumption of devices and increases earnings for device owners.

7.3 Credibility

Credibility values are mostly useful to application agents. During negotiations, they inspect the credibility (C) of devices they are negotiating with and use that credibility to decide between similar offers and to terminate negotiations with devices that don't have suitable credibility. Application owners only need to specify a single value named minCredibility (C_{min}). Application agents will then ignore devices with credibility lower than this value. When deciding between offers from two device agents, application agents take credibility into account. Application agents calculate a 'balanced price' P_B - a price with risk taken into account. P_B will be higher for devices with lower credibility. This balanced price will be used when comparing offers from devices, however, when calculating the budget or making contracts, normal offered price P will be used. P_B is calculated per formula 5.

$$P_B = P * \frac{1 - C_{min}}{C - C_{min}} \qquad (5)$$

Credibility as used in IoT Mediator Platform is a simplification of Application trustworthiness previously described in literature [33]. Application trustworthiness is quantitatively evaluated by the similarity between the application's behavior and the behavior expected by the user. It is used in the mobile or IoT context for applications deployed on devices. It consists of combination of different attributes including correctness, reliability, safeness, effectiveness, integrity, usefulness and predication [34]. In our case we are simply interested in the consistency of delivering data that device agents agreed to, so instead of having multiple attributes we have a single value $C \in [0,1]$ which simply represents the proportion of successfully delivered measurements.

More generally, this credibility mechanism is a Reputation system. Reputation can be considered as a collective measure of trustworthiness based on the referrals or ratings from members in a community [35]. Each device owner thus has a reputation/credibility that is calculated as the proportion of successful previous transactions his devices were involved in. IoT Mediator Platform credibility system satisfies the needed properties to qualify as a reputation system [36]:

1. Entities are long lived, meaning that there is always an expectation for future transactions. This is satisfied in a trivial way because the main value that device owners gain is reusing their devices and selling their data to multiple clients.

2. Ratings of current interaction are distributed to interested parties. IoT Mediator Platform observes each transaction and incorporates the result into credibility ratings which are then made available to application agents during negotiations.

3. Ratings about previous interactions impact future decisions of involved parties. Application agents use the credibility ratings as described previously in this section.

8 IMPLEMENTATION

The whole platform is implemented as a web application written in JAVA. Devices and applications can communicate with the platform using the HTTP REST paradigm. To implement RESTful web services, Jersey [7] library was used. The platform stores all of the data regarding existing devices and applications in the database. Once an application issues a negotiation request, the process goes through 3 steps:

- 1. **Preselection** - Since there could be a large number of devices, and the agent negotiation is a resource hungry process, preselection is used to reduce the time and space requirements of the actual negotiation by reducing the number of devices that are actually part of the negotiation. Preselection is based on two properties: *measurement type* and *location*. *Measurement type* refers to the physical property that the device measures. Naturally, this isn't up on negotiation since a thermometer can never measure brightness. Considering that the a majority of devices are on a fixed location, preselection can also be done so to eliminate from negotiation all devices that are outside the area that the application requires.

- 2. **Negotiation** - Once preselection is completed, for each remaining device an autonomous agent is created. This agent uses the *negotiation algorithm* specified by the device owner. It knows under which conditions it will agree to make device measurements. These device agents enter into negotiations with the application agent. Application agent keeps track of what device agents agreed to give it information and under what conditions. Negotiation flow is described in more detail in Section 6. Application agent is best described as a state machine shown in Figure 7, and in the pseudo code in Figure 1.

Algorithm 1 Application pseudo code

```
sendCFPtoAllPotentialAgents
collectResponses
for each requirements in sorted order do
  {sorted by priority or by price}
  best ← pickBestResponse
  if best is accept then
    finished;
  else
    if opponents offer final then
      if acceptable then
        send acceptMessage;
      else
        send rejectMessage;
      end if
    else
      cntOffer <- makeCounterOffer;
      send cntOffer;
    end if
  end if
end for
```

- 3. **Formalization** - Once application agent has agreed with the device agents, it sends a list of devices and negotiated terms to the *Notary agent*. Notary agent's

[7] https://jersey.java.net/

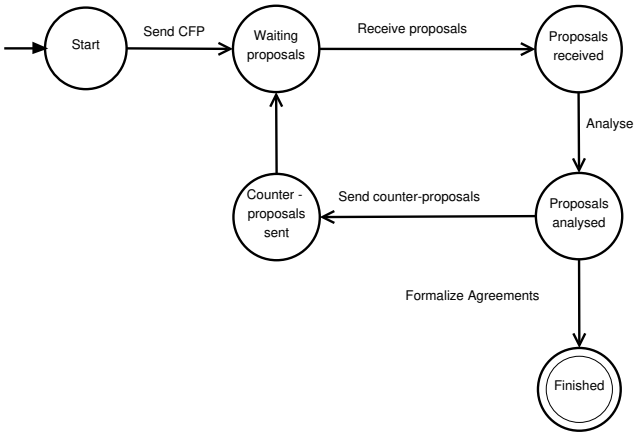

Fig. 7. Application algorithm state machine

job is to validate that information and create contracts. It first contacts all device agents so they can confirm that they indeed agreed to the listed conditions. After that the Notary agent persists the contracts in the database and notifies the web services module. At this point a HTTP answer with the made contracts is returned to the original issuer of the request.

As already stated, JADE middleware was used to implement autonomous agents and the negotiation itself. However, JADE is not well optimized when a very large number of agents is present in the system. For this reason, some parts of JADE were replaced with custom, more specialized implementations that were specifically optimized for this use-case. Most significant problem was that the DF (Directory Facilitator) agent would crash when the number of device agents present reached into thousands. Thus, we implemented a simpler and optimized version of DF agent that can function with much larger number of devices.

9 RESULTS

This platform allows users to buy and deploy Internet connected devices without having any specific application in mind. They can register their devices on the IoT Mediator Platform and specify the conditions under which they agree to give/sell the data. This way, users can earn money simply by buying and deploying IoT devices.

Application owners can quickly find the data they need for their applications. This helps them avoid large initial costs of buying and deploying devices. Also, doing this process automatically is a lot faster and cheaper than doing it manually.

TwoTariffs algorithm is an example of an algorithm that can help device and application owners to reduce power consumption. Since measuring data requires more energy than device sleeping, energy can be saved by reusing the same measurements for multiple applications. TwoTariffs enforces this behavior by making the reused measurements cheaper than new ones.

Prototype of IoT Mediator Platform described in this paper was implemented as a web application. This implementation was put under test to measure it's performance. It is important to measure the time the negotiation takes in order to asses the amount of devices the platform can handle. Tests were made using a number of simulated devices. Negotiation time depends on the number of agents that were part of the negotiation. Results are presented in Figure 8. It is important to note that this refers to the number of agents after the preselection step. So, only those agents that have the appropriate measurement type and location are a part of the negotiation.

As visible from Figure 8, negotiation time increases almost linearly with number of devices. This is due to the way negotiation is implemented. As described in Section 8, the application algorithm sends a CFP to all devices, and selects the best proposal. After that he continues negotiation with the device that made the best proposal. The duration of that negotiation is the same regardless of how many devices are registered to the platform. The majority of the time goes to the initial step of sending CFPs to preselected devices and analyzing their responses, and that is obviously linear in the number of devices.

Figure 9 shows the negotiation between an application agent using FixedBudget and a device agent using MinPrice algorithm. Current price that agents offer are shown on y-axis, and negotiation step (time) is shown on x-axis. Both agents use a tit-for-tat strategy described previously. Device target price is 20, and application budget is 20. The offering prices converge gradually towards 12 until they finally reach it in step 5. Naturally, this graph shows a situation where a deal is possible. If specified MinPrice is greater than the remaining application budget, a deal isn't achievable. Although shown only for FixedBudget-MinPrice pair, process is the same for all other pairs of device-application algorithms. Both application and device offering show a linear dynamic which is typical of tit-for-tat algorithms.

Exact numeric values of target price and application budget were chosen at random, and that is acceptable because we don't attempt to find the exact agreed price, but simply to show the dynamic of how agent's relative offered prices change during negotiations. One thing we needed to take into account when selecting the experiment parameters is that application has sufficient budget to eventually purchase the measurements.

Figure 10 shows the difference between FixedBudget and ByPriority algorithms. Total budget per application is

plotted on x-axis. Y-axis shows the number of satisfied requirements. Experiments were conducted with randomly selected devices, such that for every requirement there existed a device that is able to satisfy it. Devices used the MinPrice algorithm with randomly selected price. For each type of requirement a standard price was randomly chosen using normal distribution with mean value 20 and standard deviation 7. After that, devices each chose a minPrice from a distribution that had a mean value previously selected for that type of measurement, and stddev being 1/3 of mean value. Experiment was repeated 10 times to minimize the effect of chance. It can be seen that FixedBudget satisfies more requirements on average, the reason for that being that it prefers the cheapest requirements. ByPriority first tries to satisfy the requirements high on priority list, even if they are expensive, thus being able to satisfy a smaller number of requirements before depleting it's budget.

Similar as in previous experiment, the exact numeric values of chosen experiment parameters are not important because we only attempt to show the differences between two application algorithms. We however did choose a number of devices and prices so that the difference can be clearly observed.

Figure 11 shows the behavior (total earnings) of TwoTarrifs algorithms with different discount price settings. Total device earnings are plotted on y-axis, and the discount price is plotted on x-axis. The experiment was conducted using one device agent and 50 application agents. Device agent had normal-min price fixed at 60, and discount price varied according to x-axis, decreasing in steps of 3 from 60 to 0. Application agents were using the FixedBudget algorithm, and budget was selected at random for each application, having mean value 40 and standard deviation 30. All application were selected to require the measurement that the device was offering. Experiment was repeated 5 times, and average values are shown. First column in the graph represents the case when discount price is the same as normal price, i.e. when there is no discount offered. The graph shows that when offering the measurement at already awake times cheaper than new measurements, device can increase it's total earnings with zero additional energy. This is because a portion of applications didn't have enough budget to buy measurements at full price, but can afford it at a discount price if they are willing to adjust their timings a bit. Same is true if there are multiple devices that are competing, a device could offer a discount price with no additional cost and beat the competition. In this particular case, maximum earnings are achieved when discount price is 42.

As in previous experiments, we attempt to show the property of TwoTariffs algorithm that earnings don't increase monotonically, instead of focusing on the exact earnings in any hypothetical currency. Device target price

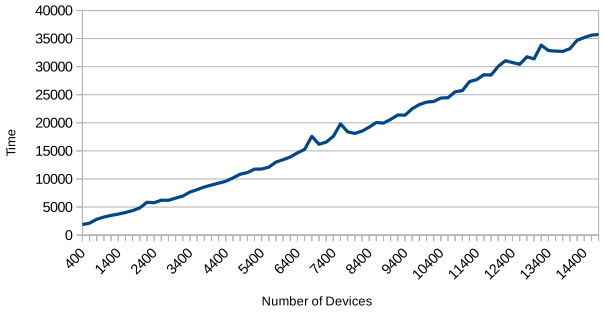

Fig. 8. Negotiation time

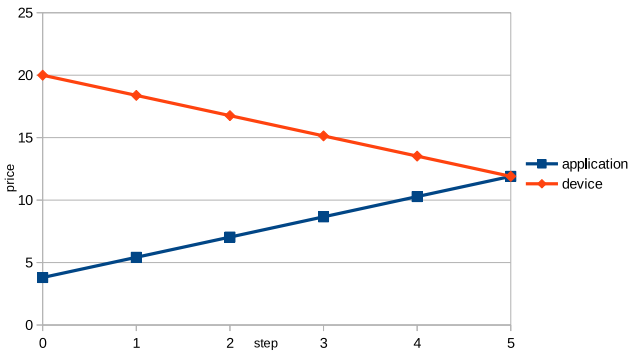

Fig. 9. Negotiation between MinPrice and FixedBudget algorithms

and applications budget distribution were chosen to clearly show this property of TwoTariffs algorithm.

Through examining the literature, we didn't find a system that allows IoT devices to sell data using a market based mechanism. Although markets using negotiating agents exist for other types of goods, they are usually made for one-time purchases of existing goods or already collected information. Special properties of IoT scenario, like many small data providers, purchasing data that has not yet been generated, and very important timing of measurements make this scenario different enough that we believe a direct comparison would not make sense. This is in-line with the main aim of this paper which is to show that negotiation agents can be used to allow selling of IoT data and handling all of the specificities of the described scenario, instead of presenting a performance improvement over any existing approach.

10 CONCLUSION AND FUTURE WORK

The IoT Mediator Platform presented in this paper is one possible solution to accomplish negotiation between Internet connected devices (in particular sensors) and applications. The negotiation itself is the most complicated

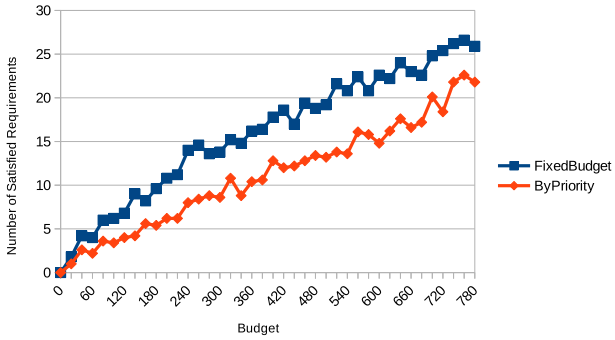

Fig. 10. Number of satisfied requirements using FixedBudget and ByPriority algorithms

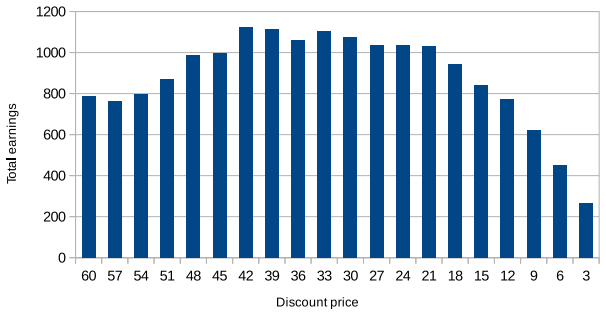

Fig. 11. Earnings of TwoTariffs in relation to it's discount price

and resource hungry part of the whole platform. Results show that negotiation complexity increases linearly with the number of agents that are negotiating. This complexity is reduced by preselection, so that a smaller number of agents takes part in the negotiation. There are a couple of predefined algorithms that users can configure to match their needs.

The analysis of different Algorithms and their comparison shows the relative strengths and weaknesses of each algorithm. ByPriority algorithm lets users select which measurements are the most important, and focus on that, but FixedBudget is able to acquire a larger number of measurements for the same amount of cash. TwoTariffs algorithm allows device owners to get an edge over competition and earn more money without draining the device battery for additional measurements.

We hope that this platform will encourage users to buy more IoT devices, simply because they can make money from the data they provide. This will lead to application owners having a richer set of data upon which to build their applications. More applications will mean more money in the system and a bigger incentive for people to connect devices. This circular process could lead to faster expansion of the IoT market.

There are still many features that we hope to add to this platform in the future. We would like to provide support for users to write the algorithms themselves, that their devices and application will use for negotiation. One way of doing this is to provide a public API against which users will be able to write their own negotiation implementations. The platform will then dynamically load the correct API implementation and let the device/application agent use it to make decisions. This API could be quite simple, there are only two crucial methods that need to be implemented. One creates a counter offer for a given offer, and the other compares a list of offers to find the best ones. However, there are inherent security concerns when executing third party code on the platform. This security issues need to be resolved before users are allowed to implement their own algorithms. One possible way of resolving them is to create a sandbox execution environment where users algorithms can be executed without endangering the rest of the system.

Security and privacy is a very important topic due to private nature of data, and proper discussion of it would be out of scope for this paper so we leave it for future work. Mechanisms that allow device owners to choose which data consumers are allowed to obtain their data or to make different requirements of different consumers are needed. For example, people might be willing to give their data to universities and environment activists almost free of charge, but not be willing to sell their data to political parties or companies looking to create users profiles or make aggressive marketing campaigns. On the other side, data consumers would like to be sure that measurements supplied by devices are genuine, and mechanisms that ensure that users are not simply providing generated fake measurements need to be devised. Further, mechanisms that prevent eavesdropping or other ways of unauthorized collection of data need to be prevented, but are not strictly related to discussed negotiation mechanisms.

Currently, IoT Mediator Platform implementation is optimized for sensors, and we would like to make it easier to use for actuators as well. This would open up a whole new range of possibilities. It is easy to imagine agents from different companies negotiating with a smart billboard on who gets to show the advertisement, etc.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for

the internet of things," *Internet Computing, IEEE*, vol. 14, no. 1, pp. 44–51, 2010.

[3] D. Raggett, "The web of things: Challenges and opportunities," *Computer*, vol. 48, no. 5, pp. 26–32, May 2015.

[4] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[5] Ericsson. (2011, February) More than 50 billion connected devices. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf

[6] S. Evdokimov, B. Fabian, S. Kunz, and N. Schoenemann, "Comparison of discovery service architectures for the internet of things," in *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, 2010, pp. 237–244.

[7] S. Mayer and D. Guinard, "An extensible discovery service for smart things," in *Proceedings of the Second International Workshop on Web of Things*. ACM, 2011, p. 7.

[8] C. Perera, A. Zaslavsky, C. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor search techniques for sensing as a service architecture for the internet of things," *Sensors Journal, IEEE*, vol. 14, no. 2, pp. 406–420, Feb 2014.

[9] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, M. J. Wooldridge, and C. Sierra, "Automated negotiation: prospects, methods and challenges," *Group Decision and Negotiation*, vol. 10, no. 2, pp. 199–215, 2001.

[10] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation decision functions for autonomous agents," *Robotics and Autonomous Systems*, vol. 24, no. 3, pp. 159–182, 1998.

[11] N. Karacapilidis and P. Moraïtis, "Intelligent agents for an artificial market system," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 592–599.

[12] A. Chavez and P. Maes, "Kasbah: An agent marketplace for buying and selling goods," in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, vol. 31. London, UK, 1996, p. 40.

[13] K. Mišura and M. Žagar, "Internet of things cloud mediator platform," *Computers in Technical Systems*, 2014.

[14] G. Norris, S. N. YoungPong, J. Q. Koenig, T. V. Larson, L. Sheppard, and J. W. Stout, "An association between fine particles and asthma emergency department visits for children in seattle." *Environmental Health Perspectives*, vol. 107, no. 6, p. 489, 1999.

[15] E. Commission. (1996) Ambient air quality assessment and management. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:1996:296:0055:0063:EN:PDF

[16] F. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*, ser. Wiley series in agent technology. John Wiley, 2007. [Online]. Available: http://books.google.hr/books?id=ZLBQAAAAMAAJ

[17] A. Fipa, "Fipa acl message structure specification," *Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00061/SC00061G.html (30.6. 2004)*, 2002.

[18] F. B. G. Caire, A. Poggi, and G. Rimassa, "Jade. a white paper," 2003.

[19] J. Soldatos, M. Serrano, and M. Hauswirth, "Convergence of utility computing with the internet-of-things," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*. IEEE, 2012, pp. 874–879.

[20] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[21] P. Jeon, J. Kim, S. Lee, C. Lee, and D.-K. Baik, "Semantic negotiation-based service framework in an m2m environment," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, vol. 2, 2011, pp. 337–340.

[22] M. Botts, G. Percivall, C. Reed, and J. Davidson, "Ogc® sensor web enablement: Overview and high level architecture," in *International conference on GeoSensor Networks*. Springer, 2006, pp. 175–190.

[23] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20–26, June 2014.

[24] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog *et al.*, "The ssn ontology of the w3c semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[25] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic sensor web," *IEEE Internet computing*, vol. 12, no. 4, pp. 78–83, 2008.

[26] X. Zheng, P. Martin, K. Brohman, and L. Da Xu, "Cloud service negotiation in internet of things environment: A mixed approach," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1506–1515, 2014.

[27] W. Hu and H. Zhu, "A methodology to enable self-organization in the internet of things based on negotiation mechanism," in *Measurement, Information and Control (MIC), 2012 International Conference on*, vol. 1. IEEE, 2012, pp. 332–336.

[28] A. McGibney, A. E. Rodríguez, and S. Rea, "Managing wireless sensor networks within iot ecosystems," in *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. IEEE, 2015, pp. 339–344.

[29] A. Keller and H. Ludwig, "The wsla framework: Specifying and monitoring service level agreements for web services," *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57–81, 2003.

[30] L. Xu and H. Weigand, "The evolution of the contract net protocol," in *Advances in Web-Age Information Management*. Springer, 2001, pp. 257–264.

[31] A. Dementyev, S. Hodges, S. Taylor, and J. Smith, "Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario," in *Wireless Symposium (IWS), 2013 IEEE International*. IEEE, 2013, pp. 1–4.

[32] J. Watson, "The tin oxide gas sensor and its applications," *Sensors and Actuators*, vol. 5, no. 1, pp. 29–42, 1984.

[33] K. Kang, Z. Pang, L. Da Xu, L. Ma, and C. Wang, "An interactive trust model for application market of the internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1516–1526, 2014.

[34] Y. Yuan and Q. Han, "A data mining based measurement method for software trustworthiness," *Chinese Journal of Electronics*, vol. 21, no. 1, pp. 13–16, 2012.

[35] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[36] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.

Krešimir Mišura is a phd student at the University of Zagreb, Faculty of electrical engineering and computing. He graduated computer science at the same University in 2012. He is currently employed at RealNetworks where he creates mobile applications for Android operating system. His interests include Internet of Things, Machine learning and mobile applications development.

Mario Žagar, Ph. D. is a tenure professor of computing at the University of Zagreb, Faculty of electrical engineering and computing. He received Dipl. ing., M. Sc. CS and Ph. D. CS degrees from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER) in 1975, 1978, 1985 respectively. In 1977 M. Žagar joined the faculty and since then has been involved in different scientific projects and educational activities. His current professional interests include: computer architectures, distributed ubiquitous and pervasive computing, open technologies and Internet related technologies. Mario Žagar is author/co-author of 5 books and more than 100 scientific/professional journal and conference papers.

AUTHORS' ADDRESSES

Krešimir Mišura, Mag. Ing.
Prof. Mario Žagar, Ph. D.,
University of Zagreb,
Faculty of Electrical Engineering and Computing,
Department of Control and Computer Engineering,
Unska 3, HR-10000 Zagreb, Croatia
e-mail: kresimir.misura@fer.hr, mario.zagar@fer.hr

Received: 2015-01-26
Accepted: 2016-09-16