Advance Internet of Things

Lei Zhang, Nathalie Mitton

Advanced Internet of Things

Lei Zhang, Nathalie Mitton
INRIA Lille - Nord Europe, Univ Lille Nord de France
USTL, CNRS UMR 8022, LIFL, France
Villeneuve d'Ascq, France
Email: {lei.zhang, nathalie.mitton}@inria.fr

***Abstract*—The Internet of Things (IoT) is a network of Internet-enabled objects, whose original purpose would be to interconnect all things in our daily life to build an always-connected world. However, most of studies in the current IoT scientific community only focus on the radio-frequency identification (RFID) and wireless sensor network (WSN) based objects and lose the generality features endowed by the original definition of IoT. Furthermore, the emergence and proliferation of smart objects have been significantly changing our daily lives. It has been becoming evident that the objects should far beyond only "be identified and interconnected", but can also be controlled in an intelligent and transparent way independent of third party object (user) profiles and space & time span. In this paper, we proposes a standardization scheme for a new paradigm: Advanced Internet of Things (AIoT), which is based on our proposed Unified Object Description Language (UODL) and allows to identify and interconnect every object and event with a standard format, and makes it easier and flexible for the third party control and management by integrating multiple services issued from cloud computing. The purpose of our proposed AIoT scheme is to build a smart world of always-on, always-awareness, always-connected, always-controllable, and establish an "intelligent networking" based relationship among the objects, service suppliers and the third party users. In the scope of AIoT, all the objects are transparent across the networks and can be identified and controlled (with security guarantees) via a standard prototype anytime and anywhere.**

***Keywords*-Internet of things; Internet of services; cloud computing; network based services**

I. Introduction

We are entering into a new age where the number of "things" is in explosive growth. The conception of "Internet of Things"[1] prompts a novel pervasive vision and opens a paradigm that makes all the objects interconnected, responsive, adaptive and omnipresent around our lives. Every object is expected to be endowed with an always-connected capability guaranteed by heterogeneous wireless technologies, heralding the dawn of a new era in which scalable implementations of "Machine to Machine" (include Machine-to-Machine-to-Human M2M2H) [2] services will be largely deployed. Such smart-object based networks will bring significant changes across our daily lives and a range of industrial manufacturing sectors in the following five to ten years [3], [4]. The key technological drivers behind the Internet of Things should be a mixture of heterogeneous technologies across the wireless communication, pervasive embedded systems, Internet services and security issues. In the current IoT scientific community, a set of issues is still pending and needs to be carefully addressed. Firstly, an open and global standardization of IoT is still in infancy or remains fragmented, which prevents the IoT related deployments from expanding to a global scale [5], [6], [7], [8]. Secondly, most of the research work only focus on the RFID and WSN based object [24], [25], [26], [32], and lose the generality features originally defined in the scope of IoT. Thirdly, most proposed solutions in the current IoT community are still constrained by its traditional purpose that is simply "interconnecting and tracking all the objects" [21], [30], [31], we are in a world where the smart objects will be perfusing in every corner around our lives, and those intelligent objects should be completely controllable independent of the user[1] profiles and time & space span instead of only being"Inventoried and Interconnected".

Keeping in mind the above mentioned deficiencies in the scope of IoT, in this paper, we propose a standardization scheme for a new paradigm: Advanced Internet of Things (AIoT), which results from a combination work of standardization, architecture design, application deployment as well as its business model. *Firstly*, AIoT scheme is based on our proposed **Unified Object Description Language (UODL)** which is used to "describe" each object, including its standardized ID, its properties (related information) and a description of the related services and controlling issues related to the object. UODL allows to 1) easily **identify** and **interconnect every object** with a **standard** format; 2) make all the information related to the objects (i.e. properties, available services) **transparent** across the networks; 3) make it easier and flexible for the **third party control and management** via networks. *Secondly*, AIoT scheme is based on large scale **distributed architecture**, and all the interactions among the objects (including human, machine, events, etc.) are based on the service-enabled networks,

[1]In the context of our proposed AIoT scheme, we define that *user* represents any third party objects or end users who want to get the target object information or control and run the applications related to the target object.

which allows the service supplier [2] to deploy the object-related services more flexibly. *Thirdly*, we propose a **novel business model** in the scope of AIoT that brings forward a new relationship between the object and its related service supplier, which offers a more efficient and flexible way for the service management.

Meanwhile, AIoT scheme is characterized by the following four main features, which is expected to drive AIoT architecture to a large scale deployment and conform to the future networking. 1) **Cloud computing**: AIoT keeps in mind that all the complex belongs to cloud side (i.e. service supplier side) instead of local side (i.e. end user side), and supports an optimization management on the computing resource. 2) **Compatibility**: A seamless update from the current network stack and existing service systems to AIoT architecture is supported. 3) **C2C mode** (customer-to-customer, namely "customer develops for custom" or "everyone can be a developer", which was originally adopted as Android and Apple business mode): This conception is also employed by AIoT scheme to make "everyone can be a service supplier", which means that anyone can flexibly distribute his developed services (i.e. applications related to an object) anywhere with a network access, and thanks to our proposed AIoT standard prototype, all the services related to each object can be easily accessed and triggered by users via the networks. 4) **User Experience**: In the context of AIoT, user is able to, via an unique standard AIoT interface, be aware of all the information of every object, and triggers all the services related to every object in a way of transparency and simplicity independent of operation terminals and time & space span.

To the best of our knowledge, we are the first, in the context of IoT community, to propose a standardization work which covers the following two principal axis simultaneously: 1) Horizontal axis: our scheme weaves a cyberspace which takes consideration of **all the "things"** (i.e. across the RFID and WSN based object, smart object, abstract object, event, etc.), and makes them easier and flexible for the third party control and management. 2) Vertical axis, AIoT scheme combines a series of work span from **standardization, architecture design,** to **application deployment** as well as **its business model**.

This paper is organized as follows: Section II gives some related work, Section III introduces our proposed Unified Object Description Language. Section IV shows the AIoT architecture. Application deployment and the business model are introduced in Section V. Section VI gives a discussion on the AIoT deployments, then conclusion will be finally given in Section VII.

[2]We define that ***service supplier*** represents the provider of the services and applications related to the target object (i.e. manufacturer of the target object, who is able to control it in distance and run its related applications.)

II. Related work

IoT is originally promoted with the electronic product code (EPC) technology [22] developed by Auto-ID Center of MIT. A great number of works in the IoT community focus on the EPCGlobal based RFID and WSN architecture. In [23], authors propose an universal resource addressing system which can support any Product Code Standard. Authors in [24] give a review of current RFID technology and the IoT deployments based on EPC stack. In [25], authors introduce several promising technologies employed in the IoT scope and investigate an address mapping mechanism, which is compatible with existing RFID framework, and enables the communication between heterogeneous RFID systems. In [26], authors present the fundamental concepts and applications of the EPC Network, as well as its functionality of data exchange for IoT applications. Authors in [9] propose a new concept, based on the local modifications to the RFID readers, to enable the IPV6 compatibility to passive tags, and then make the RFID tags communicate as an end node over the Internet. In [27], authors give a review on the current IoT status and propose a novel heterogeneous and self optimizing sensor network management system for IoT applications to interconnect heterogeneous devices and keep in sync with the existing Internet architecture. Authors in [28] propose an IoT Gateway system, based on protocol conversion and control functionalities, which allows facilitating the seamless integration of wireless sensor networks and mobile communication networks in IoT applications. Furthermore, a set of work address on the global architecture of IoT, [29] gives a semantic analysis for IoT. In [20], authors propose, extended from current IoT stack, a 5 layer-based architecture of IoT to conform the future IoT applications. In [30], authors show an IoT architecture which supports ubiquitous services on an end-to-end basis, and authors in [31] address some essential issues and technologies of the IoT on its architecture, interoperability, as well as the application service categories. In [21], based on an analysis on the IoT basic attributes and features, a descriptive models has been proposed to integrate the Internet application into the IoT scope. Authors in [33] give a gloabal view on the IoT involved technologies, standards and the related deployements. However, we argue that few work in the current IoT community takes into account both all "things" (i.e. non-RFID and WSN based object, abstract object, event,etc.) and proposing a framework which includes the standardization work, architecture design, application deployment as well as its business model.

III. Unified Object Description Language (UODL)

In the scope of AIoT, all the information related to the objects should be transparent across the networks, this information includes the standarlized ID of the object; the related data (properties) of the object; and a description of

the related services and the controlling issues related to the object. We propose an Unified Object Description Language (UODL) which allows to "describe" these informations and offer a unified interface to communicate between service suppliers and users

UODL is based on XML language [10] and includes four principal parts to describe an object: *AIoT ID, Standard Properties Description*, *Standard Actions Description*, *Standard ActionTriggers Description*. Figure 4 shows an example which represents the global structure of UODL.

A. AIoT ID

In the context of AIoT, every object should be assigned with an standardized ID (AIoT ID). This ID is similar to *urn* [11] and *EPC tag* format [12], and can be separated into five fields by colon ":". This standardized ID is normally assigned by the service supplier or the owner of the object. In the following explication, we take an example that the target object refers to an aircondition and its manufacturer (service supplier) is Samsung.

- The first field represents the domain information of the concerning object. If the object is unique and makes sense in the global context where the AIoT architecture is deployed, the first field must be filled with "urn" (case-insensitive). If the object just makes sense in a local network and not in the global context, the first field must be filled with "local" (case-insensitive).

- The second field represents the type of the ID, which is one of the following (case-insensitive): "sid", "hex", "pattern", "asid", "ahex".

- *"sid": Standard ID (SID) is normally named in the format of String by the service supplier / owner of the object, and is the most common format used in AIoT architecture. i.e. "urn:sid:fr:samsung:AirCondition_X100.N2341" represents an aircondition (model X100) with the serial number N2341 manufactured by Samsung in France.*
- *"hex": hexadecimal format of the "sid". The hex format is only used to be treated directly by machine.*
- *"pattern" is used for filtering and grouping the objects, which adopts the conception defined in EPC datatype Pattern Syntax [12] and is principally used for database management. Character "X" and "*" can be respectively used in the third, fourth and fifth field for Grouping and Filtering operations.*
- *"asid", "ahex" represent alias ID formats for the corresponding object. In AIoT scheme, one object can correspond to multiple ID, however, only one main ID (the second field of main ID must be filled with "sid") makes sense in the whole context where AIoT deploys, and alias ID corresponds to the object that only makes sense for the local end users. i.e. For the aircondition (its main ID "urn:sid:fr:samsung:AirCondition_X100.N2341"), an alias ID such like 'urn:asid:MyHomeAutomation:AirCondition:BedRoom" can be named to be compatible with the home-automation network at the user side. When the user uses this alias name, the system will automatically point to the corresponding main ID to communicate with other nodes across the AIoT networks.*

- The third field (when the second field is not in "hex" or "ahex" format) represents the first classification of the corresponding object, the first classification can be freely assigned by its service supplier or its owner. Normally it refers to (but not constrained to) the country prefix (for global deployment) or the name of the principal service supplier of the object. i.e. "urn:sid:fr:samsung:AirCondition_X100.N2341" represents that the manufacturer or the service supplier of the concerning object is in France; Another example, "urn:sid:epc:id:sgtin.003700.03043.12334" represents that the service supplier of the object is EPC, which offers the EPC ONS [13] lookup services. The standard ID in the context of AIoT can be easily translated to EPC ONS compliant format and thus is compatible with EPC related standards [14].

- The fourth field represents the second classification of the corresponding object, which is freely defined by service supplier and owner of the concerning object. Normally this field refers to (but not constrained to) a service supplier.

- The fifth field represents the data field that can be separated by point ".", which allows further classifications and end identification. i.e. the fifth field of "urn:sid:fr:samsung:AirCondition_X100.N2341" includes a further classification (the model of the aircondition "AirCondition_X100") and an end identification (the manufacture serial number "N2341"), such standard ID format offers an unique identification of the object in the global context where the AIoT architecture is deployed. Pattern syntax (Filtering and Grouping) is also available in fifth field which can be separated by point "." i.e. "urn:sid:fr:samsung:AirCondition_X100.*" represents all the airconditions with the model X100 manufactured by Samsung in France.

This proposed ID format also allows to identify an abstract object and event, whose purpose is to offer a standard interface to interconnect every "thing" (i.e. EPC compliant objects, smart objects, abstract objects, etc.) around our lives, and this ID standardization is considered as a cornerstone on which our proposed architecture is built.

B. Standard Properties Description

In the context of IoT, an object does not make sense if it's just "connected", the users can be more interested in its properties and its related data. In AIoT scheme, we suppose that all the related data (*properties*) of the object should be transparent across the networks. We propose a *Standard Properties Description* (Figure 4, example in lines

5-22) which is based on XML language to "describe" the object properties.

The <PropertyName> represents the name of the property, the <Value> represents the value corresponding to the <PropertyName>. The field <PropertyName> is normally defined by service supplier or the owner of the object. The field <Value> can be updated by service supplier, the owner of the object, or other third party user.

C. Standard Actions Description

Nowadays, with the maturation of IPv6, more and more IP-enabled smart objects will be perfusing in every corner around our lives, which promotes a new era of intelligent remote control. In the context of AIoT, objects are supposed to be totally controllable anytime and anywhere, and they can be controlled via networks independently of the user profiles, meanwhile, the services and applications related to the objects can be easily accessed via networks by the third party object in an intelligent and transparent way. ***Actions*** in our AIoT scheme means: 1) Control an object. 2) Run services or applications related to an object. Specially, we would like to make clear the conception of *"object controlling"*. First, we argue that controlling an intelligent object is relative simple and not as complicated as managing a complex system that requires significant interactions with end users, which means an object can be controlled in the following way: an user sends commands and several operation parameters in standard format to the service supplier side, and then triggers the service supplier to pilot the remote object or run the related services. In the context of AIoT, the communication between the user and the service supplier is based on UODL which is a light-weighted XML resolve, which can be deployed in any smart device (including resource limited device) like laptop, smart phone, Google Chrome OS, etc.

Figure 4 (lines 23-52) represents an example of the format of the *Standard Actions Description* which represents the corresponding services and applications related to an object. Multiple <*Action*> can be defined for one object, an *Action* includes 5 principal domains: 1) <*ActionName*>; 2) <*LaunchTime*> (defined by users and represents "when to launch this application?"); 3) <*Description*> (a description of the *Action*, users will be aware of it before triggering the corresponding action). 4) <*Parameters*> is used to guide the users how to configure the right parameters to trigger the corresponding *Action*, multiple <*Parameter*> might be required to trigger one *Action.* <*ParaName*> represents name for a parameter, and this parameter should be in a type that is defined in <Type>, and this parameter can belong to one of the values defined in <*BelongTo*>. <*Indication*> is the indication of the parameter which is defined by service supplier and is shown at the user side (i.e. if "Temperature setting:" is set for the field <*Indication*>, this string will be shown on the auto-generated graphical interface at the user side, and indicates the user to input a value to set the temperature, please also see section IV-C and IV-D). <*Value*> represents the value of this parameter and is inputed by user in terms of the value set in <*Indication*> , this value will be later sent to the service supplier side to trigger the related services. 5) <*ActionResults*> is used to feedback some operation results from service supplier to user.

D. Standard ActionTriggers Description

Standard ActionTriggers Description is proposed to support scalable implementations of "Machine to Machine" architecture in the AIoT context. *Standard ActionTriggers Description* is defined by service supplier and contains multiple *ActionTrigger* which allow triggering multiple *Actions* when the trigger conditions are satisfied (see Figure 4, line 53-82).

Each <*ActionTrigger*> includes 2 principal parts: <*condition*> and <*triggers*>. The field <*condition*> is defined similar to the programming language, the following operations are allowed (==, !=, &&, ||). For example, <*Condition*> *($status$=="ERROR")&& ($notified$=="NO")* </*Condition*> , the *$status$* and *$notified$* refers to the values corresponding to the property name *status* and *notified*. This trigger condition means: if the value of *status* equals to "ERROR" and the value of *notified* equals to "NO", then the defined actions (defined in <*triggers*>) will be triggered. <*triggers*> can include multiple <*trigger*>*,* and every <*trigger*> contains an *Action* which is introduced in section III-C.

E. Use of UODL

UODL serves as a standardized communication language exchanged between service supplier and user, the field <*TYPE*> can be set as one of the follows according to the different purpose of using UODL: *(ADD; RESULT_ADD; DELETE; RESULT_DELETE; UPDATE; RESULT_UPDATE; REQUEST; RESULT_REQUEST; ACTION; RESULT_ACTION).*

ADD: the purpose of the current operation is to add a description (including *AIoT ID, Properties, Actions and ActionTriggers*) for a new object in an Object Server where stores all the informations related to objects at the service supplier side (more details in section IV-B). This operation is normally concerned by service suppliers. *RESULT_ADD* is the result indicating whether this operation is successful.

DELETE: the purpose of the current operation is to delete a description of an object in the Object Server. This operation is normally concerned by service suppliers. *RESULT_DELETE* is the result indicating whether this operation is successful.

UPDATE: the purpose of the current operation is to update a description of an object in the Object Server, values of multiple fields can be added, modified and updated. This

operation is normally concerned by service suppliers or users. An example is given in Figure 4. *RESULT_UPDATE* is the result indicating whether this operation is successful.

REQUEST: the purpose of the current operation is to request a full description (or values of several specified fields in terms of the request specification) of an object from the Object Server. This operation is normally concerned by authorized users who want to get the object information or control the related object. *RESULT_REQUEST* returns a full description (or the values of several specified fields) of an object.

ACTION : the purpose of the current operation is to send the *Action* parameters to the service supplier, and demand to launch one or multiple specified services or applications related to the corresponding object. This operation is normally concerned by authorized users. *RESULT_ACTION* is the action results returned to users from service supplier side.

A field <*Security*> which includes security control is also introduced in UODL. i.e. User can access to the corresponding Object Server at the service supplier side via a login/password control.

IV. AIoT ARCHITECTURE

AIoT architecture is based on a service-enabled network, which comprises 4 principal components: Advanced Object Naming Service (AONS); Service Supplier Domain (SSD); User (with AIoT Standard Interface) and Target object (Figure 1). The interactions between users and service suppliers are mainly achieved by the unified standardized specification UODL.

We firstly give a detailed introduction on the AONS and SSD, then a global view of how AIoT architecture works will be shown.

A. Advanced Object Naming Service (AONS)

Similar to ONS [13] and DNS [15], AONS offers a mapping service between object ID and the IP address of its Service Supplier server. In the context of AIoT, all the information (*Properties, Actions, ActionTriggers*) related to objects are stored in Object Servers in SSD (more details about SSD is in section IV-B), and the role of AONS is to "guide" the user to find the IP address of the corresponding Object Server in SSD which contains all the information of a specific object with its SID.

Similar to an IP address that can be logically recognized as consisting of two parts: the network prefix and the host identifier. A SID can also be separated into two parts, the first part (public part) makes sense in the public AONS and serves as "routing" to the access point of the corresponding SSD, which is normally an IP address of a Local AONS (LA) of the corresponding SSD. The second part (local part) of SID only makes sense in the LA, which records mappings between an object SID and the IP addresses

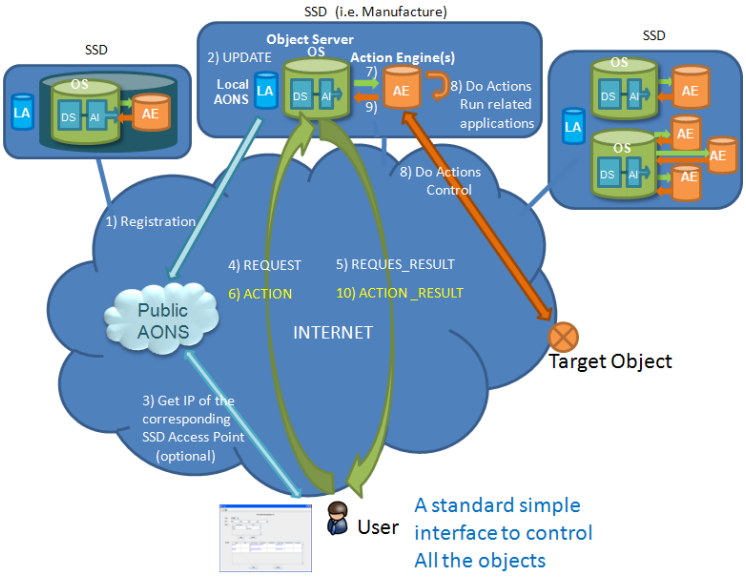

Figure 1. AIoT architecture

of the Object Servers which contains information of the corresponding object. Therefore, the procedure of AONS can be divided into 2 steps: 1) The user firstly sends a SID to public AONS, the architecture of AONS is similar to DNS, which is illustrated in Figure 2, such tree topology allows tracing the IP address corresponding to the end child of the public part of SID. (i.e. if a user send a SID: urn:sid:fr:samsung:AirCondition_X100.N2341 to the public AONS, the public AONS tries to trace the IP address until the domain *root->fr->samsung*, and it can not find any child under the domain *samsung* (Figure 2), then it returns the user the IP address of the access point that corresponds to the public part of SID: *urn:sid:fr:samsung*, which represents the IP address of the Local AONS of the corresponding samsung SSD). 2) Then the user sends the SID to the IP address of the Local AONS in the corresponding SSD (see in Figure 1), which offers a similar mapping service between the local part of SID and the IP address of the corresponding Object Server. (i.e. the Local ANOS returns user the IP address of the local Object Server corresponding to the object whose local part of SID is *AirCondition_X100.N2341*). Then similar to the DNS, the mapping between the target Object Server IP address and the SID will be stored in local cache, user is able to access to the corresponding Object Server directly and request object information without re-processing the above procedure later.

B. Service Supplier Domain (SSD)

The role of Service Supplier Domains in AIoT is: 1) Offers a flexible way for the user to access and request all the information related to objects. 2) Run the applications and services related to objects with the requests from users.

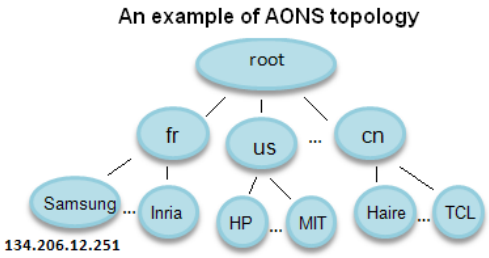

Figure 2. Tree topology of AONS

SSD consists of 3 principal components: Local ANOS (LA); Action Engine (AE); Object Server (OS) (Figure 1).

The role of LA is to mapping the object to the corresponding OS in service supplier local side, and it has been discussed in the previous section IV-A;

The role of AE is to *do Actions*, it contains a set of services (i.e. applications, programs, RMI servers [16], etc.) related to the objects, for example, to control remote objects, run an application related to objects for users. We emphasized that AE can be some already-existing independent services or third part developed services, and it can be flexibly integrated in AIoT architecture via Action Interface, therefore, we argue that our proposed AIoT architecture is completely compatible with existing service systems.

OS comprises two components: 1) Data Service (DS) which stocks all the information (*AIoT ID, Properties, Actions, ActionTriggers*) of objects, and handles the following operations which are defined in section III-E: *ADD; DELETE; UPDATA; REQUEST*. 2) Action Interface (AI) which allows handling the *ACTION* operation and triggering the corresponding AE to do Actions. Specially, it de-encapsulates *ACTION* specification; retrieves the values of the fields <*ID*>, <*ActionName*>, <*LaunchTime*> and <*Parameters*> defined in the *action*s specifications; and then triggers the AE to run the corresponding applications and services in terms of these retrived values. AI also allows encapsulating *ActionResults* and return them to users.

The configuration of the 3 principal components is flexible according to different deployment scenarios, i.e. AE and OS can be on the same machine, one OS can be connected directly with multiple AE, etc. Thanks to the SSD, we push AIoT to a scope that enables both Object based and Service based Networks.

C. AIoT Standard Interface (ASI)

A graphical based AIoT Standard Interface (ASI) (Figure 3) has been developed to offer a flexible and convenient way for the users and servicer suppliers to manage and interact with AIoT components. ASI allows to receive UODL based specifications, retrieve the useful information and show them through a friendly graphical interface. Meanwhile,

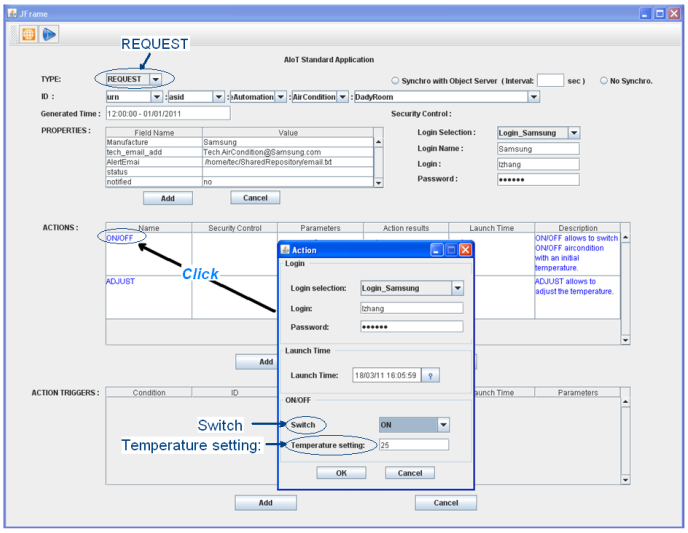

Figure 3. AIoT Standard Interface

ASI also allows to translate the user's inputs to UODL based specifications and send them to the remote servers. Concretely, service supplier can use ASI to easily manage (i.e. through ADD/DELETE/UPDATA based specification) the objects information with a friendly interface, and users can use this simple ASI to easily get the information of every object and trigger all the authorized services related to every object across the networks (i.e. through REQUEST/ACTION based specification) .

D. Procedure of launching AIoT services

The procedure of launching AIoT services comprises the operations at both users and servicer suppliers' sides. The protocols involved for the UODL based communication between the user and service supplier is open and can be dynamically configured in different scenarios. In our case, we use TCP/SSL [17] to guarantee the security for the communication between users and service suppliers.

If a user is willing to trigger a service related to an object, the complete procedure should include 10 principal steps:

Step 1) Service Supplier should firstly register in the public AONS the mapping between the IP address of SSD access point and the public part of SID (i.e. manufacture domain), and an implementation of Local AONS should be deployed which mapping an object SID to the corresponding IP address of specific OS. The purpose of this AONS deployments is to, via the SID of an object, "guide" an user to find the IP address of the corresponding Object Server.

Step 2) Service Supplier should add/update (via UODL based specification <*TYPE*>=*ADD* or *UPDATE*, see section III) all the information of the target object in the corresponding OS in the SSD. Figure 4 represents an example of the *UPDATE* specification. This UODL based specification can be i) edited directly in language UODL by Service

Supplier, or ii) edited through the ASI, which offers a friendly interface and allows generating automatically the UODL based specification.

Step 3) At user side, user requests all the information of the target object from the corresponding OS. User should first check whether the mapping between the SID and the IP address of the correspond OS is in local cache, if it's not found, run the Advanced Object Naming Service which is discussed in section IV-A to find the IP address of the corresponding OS.

Step 4) User sends a REQUEST to the corresponding OS in the SSD to retrive the information of the target object. This operation can be simply done via the ASI by inputting the *sid* (or *asid*, see section III-A) and choosing the TYPE as "REQUEST", and possibly inputting a login/password security control.

Step 5) The corresponding OS verifies the security control and returns the user a detailed description (including the *Properties* and the *Actions* descriptions via UODL specification) of the target object. Similarly, ASI is able to intelligently handle the received UODL specification, retrieves the useful information and shows them to user via a graphical friendly interface (Figure 3). Until now, user is aware of all the information of the target object, including the available services related to the object, which are defined by the service supplier. In the example illustrated in Figure 3, we developed in AI and AE an emulator to control an intelligent aircondition in distance, the two available services are respectively: *ON/OFF*, a service which allows a remote user to switch "on" or "off" the aircondition with an initial temperature, and *ADJUST* allows a user to adjust the temperature remotely for the aircondition.

Step 6) The user is aware of all the information of the target object, and if the user wants to launch an available service, he only needs to click the Action Name on the ASI and input the necessary parameters. Indeed, ASI is able to analyze the received UODL specification resulted from step 5, and automatically generate a graphical interface according to several specific fields set in <*LaunchTime*> and <*Parameters*> (also see section III-C). Such auto-generated graphical interface allows user to input the necessary parameters (Action graphical interface in Figure 3), and generate automatically a UODL specification with <*TYPE*>=*ACTION* which contains these inputted parameters, and send such specification to the corresponding OS in the target SSD to trigger the related services.

Step 7) When OS receives the specification, the AI will de-encapsulate this *ACTION* specification; retrieve the values of <*ID*>, <*ActionName*>, <*LaunchTime*> and action <*Parameters*>; and then trigger the related AE to do the corresponding *Actions* at the *LaunchTime* defined by user.

Step 8) AE runs the *Actions*, the *Actions* represents i) Control the remote object, or ii) Run the corresponding applications and services. In our testing example, AE can automatically locate the network address of the target object with its standard ID via some specific active-discovery and address/port translation protocols (in case that the object is in a NAT/NAPT behind a router [18], [19]), and is able to remotely control the target object. Note that this part is developed independently by the service supplier, it can be already-existing services and independent to the AIoT scheme.

Step 9) The action results are transmitted to the corresponding AI in OS.

Step 10) AI then encapsulates the action results with a UODL specification with <*TYPE*>=*RESULT_ACTION* ,and sends it to user. At the user side, the ASI is able to resolve such specification and show the action results with a graphical interface thanks to the fields setting of <ResultName>.

Furthermore, users, service suppliers or even the smart object itself are able to update the *properties* of the smart object in the OS (via UODL based specification <*TYPE*>=*UPDATE* and specific authorized permission) anytime according to the scenario setting. These information will then be transparent across the networks. AIoT services can also be triggered automatically by *ActionTriggers* (see section III-D), every time the *UPDATE* operation is done, OS will check the new values of the *properties* and verify whether the trigger conditions are satisfied.

Such procedure enables Internet of services for specific connected objects. We argue that all the communication between the user and service supplier is based on a light-weighted UODL, and we push all the complex to the service supplier side, user is able to control every connected object via an unique simple and terminal-independent standard interface ASI anywhere and anytime.

V. Application deployment and the business model

AIoT is based on large scale distributed architecture and can be deployed across our daily lives and a range of industrial manufacturing sector, two typical scenarios are mainly characterized in the context of AIoT. Firstly, in most cases, a service supplier can be a manufacturer, or the service provider of a connected object, which offers the services to control and run the related applications of the target object with the requirements from users, such like control home appliances in distance, the OS and AE are at the service supplier side. In the second scenario, OS and AE can be dynamically distributed across the networks, they can be installed in any network-enabled servers in terms of specific application configurations (i.e. service supplier local servers or third party public cloud servers) and the objects, related services, users and service suppliers are interconnected via Internet. This scenario offers a flexible way for the service deployments, for example SME (Small and Medium Enterprises) can store the related database and services on the low-cost third party public servers.

Specially, it enables a C2C mode which is "everyone can be a service supplier", an individual developer can flexibly develop and upload his applications and services (with a Standard Actions Description based UODL) to any network-enabled servers, and the information will be transparent across the networks, any authorized user is able to easily access and apply the related services with a corresponding service SID via the AIoT standard Interface which is able to auto-generate an interactive graphical interface for the corresponding application.

Meanwhile, we propose a novel relationship among users, object and its service supplier. The traditional interactive relationship is very dependent and inefficient, normally, service suppliers have to install the related applications and services at the user side to make the target objects controllable by the users, which can be very costly. In the AIoT scheme, the three principal roles are relatively separated: Service supplier takes charge of the complex (i.e. develop and update the related services, applications, pilots, etc.) at the cloud side (i.e. service supplier side or third party public cloud servers), and supports an efficient central management for the related services, service supplier is unaware of users'information and keeps relatively independent to the users. Meanwhile, no technical issues (i.e. related applications, services) are concerned at the users side, the only thing an user should do is to manipulate the terminal-independent interface ASI with the SID of the target objects (and possibly security control such as login/password), and to access and trigger all the authorized services (or control) of every connected object via the networks. Furthermore, AIoT is based on the service-enabled networks, the target objects can be independently deployed anywhere with a network access, which offers a more efficient and flexible way for application deployments.

VI. Discussion

The core of AIoT architecture is networks. Objects and services are distributed across the networks and can be easily accessible by the users via ASI. A valuable question to be discussed is raised as follows: why the users should control objects via the remote OS at the service supplier side instead of controlling the objects directly, specially in the cases when the objects are close to the users. 5 reasons are listed as follows to answer the questions, they are also considered as the benefits of the proposed AIoT architecture:

1) We are in the transition to the Next Generation Network (NGN) which is characterized by heterogeneous promising communication technologies, and is supposed to offer a high broadband transmission as well as the Quality of Service guarantee (i.e. low packet rate and low end to end delay). Users will be experiencing ignored delay and high reliability to control the target objects in the AIoT context.

2) Such architecture offers the service suppliers an efficient central management for the services. Service supplier can flexibly add, update and extend services and applications related to different objects in SSD, and nothing should be done at user side, and users can always track and trigger the latest updated services via the operation *REQUEST/ACTION* based on UODL. Furthermore, AIoT supports the cloudy computing deployments, which allows the service supplier to dynamically configure the computing/server resources (i.e. data and services can be flexibly deployed in any network enabled servers in terms of specific scenarios)

3) If users want to control different objects directly, they have to probably install different applications and services on users' local machines, and these applications and services are normally terminal-dependent and are not compatible with several resource-limited devices. In our proposed AIoT scheme, all the complex belongs to the cloud side, and simplicity leaves at user side. users are able to get all the information of every object and trigger all the authorized services via one unique and intelligent interface ASI. Specially, ASI is based on the light-weighted UODL and can be deployed on any smart device (including resource limited device) like laptop, smart phone, Google Chrome OS, etc. Therefore, users are able to control every connected object anywhere and anytime. Furthermore, users are not supposed to install N different applications/softwares to control N different objects, all the controling/service operations concerning to different objects are via the unique interface ASI, therefore, in a point of economics view, no costs are required for service suppliers to develop and install the related applications at the user side, meanwhile, the object-controling and the related services triggering are independent of terminal profile and space & time span.

4) Such architecture offers a simple and intelligent way to trigger the M2M services. It's enough to configure the objects information (*ActionTriggers*) in OS and update the independent AE to enable new M2M services, and no additional infrastructures and applications are required to be installed at user side.

5) The services may not refer to controlling an object, but running an application, and the service supplier may not refer to a manufacturer, but an individual developer. All these applications are distributed across the network, and are able to flexibly accessed and launched by users via ASI. In the AIoT scheme, users are no longer constrained in a local domain between user<=>object, but have an global view across the whole objects and services based Internet.

Of course, we argue that, in the context of AIoT, service supplier can dynamically configure the way of how to deploy the services and applications for different objects, the traditional way (user <=> object mode) is not strictly excluded, it can work with other AIoT compliant implementations. For example, in an AIoT deployment context, according to different specific scenarios, certain objects can be more suitable to be controlled directly by users regarding of the security consideration or users' requirements, and

other objects are deployed to conform the AIoT standardized scheme in a view of flexibility, economics and efficiency.

Another important issue in the AIoT remains the security guarantee, we apply currently the most common used *login/-password* control and TCP/SSL communication to access and trigger services for the users. The involved techniques of security guarantee are totally open and can be flexibly configured in practical deployments. In our future work, we will investigate the advanced security issues and deploy such AIoT architecture in large scale cooperating with our industrial partners.

VII. Conclusion

In this paper, we propose a new paradigm: Advanced Internet of Things, which is based on Unified Object Description Language, and allows not only to interconnect every object, but also offers a transparent and intelligent way for users to control it or trigger the related services issued from cloud computing. In the scope of AIoT, a new relationship between service suppliers and users is introduced, which results in significant benefits regarding of the efficient service management, economic implementation, as well as the enhanced user experience. Furthermore, our proposed architecture is completely compatible with the existing service systems, which is able to easily drive its deployment into a large scale. As far as we know, we are the first to propose such global IoT architecture which comprises standardization, architecture design, application deployment as well as its business model. We believe that our work will open a new era and help to put forward the IoT standardization procedure to build an intelligent world of always-on, always-awareness, always-connected, always-controllable.

Acknowledgment

This work was partially supported by a grant from CPER Nord-Pas-de-Calais/FEDER Campus Intelligence Ambiante, European FP7 ICT IP Project Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications (ASPIRE).

Appendix

```
<TYPE>UPDATE</TYPE>
<ID>urn:sid:fr:samsung:AirCondition_X100.N2341</ID>
<GeneratedTime>120000010111</GeneratedTime >
<Security> </Security>
<Properties>
      <Property>
            <PropertyName>status</PropertyName>
            <Value> </Value>
       </Property>
      <Property>
            <PropertyName>tech_tmail_addr</PropertyName>
            <Value> Tech.AirCondition@Samsung.com</Value>
      </Property>
      <Property>
            <PropertyName>AlertEmail</PropertyName>
            <Value>/home/tec/SharedRepository/email.txt</Value>
      </Property>
      <Property>
            <PropertyName>notified</PropertyName>
            <Value></Value>
      </Property>
</Properties >
<Actions>
   <Action>
    <ActionName>ON/OFF</ActionName>
    <LaunchTime> </LaunchTime>
    <Description>ON/OFF is used to ¡ </Description>
      <Parameters>
         <Parameter>
            <ParaName>Para1</ParaName>
            <Type>String</Type>
            <BelongTo>("ON", "OFF")</BelongTo>
            <Indication>Switch</Indication>
            <Value></Value>
         </Parameter>
         <Parameter>
            <ParaName>Para2</ParaName>
            <Type>float||int</Type>
            <BelongTo> </BelongTo>
            <Indication>Temperature setting:</Indication>
            <Value></Value>
         </Parameter>
    </Parameters>
    <ActionResults>
          <ActionResult>
            <ResultName>Result_1</ResultName>
            <Indication>Operation Result</Indication >
            <Value></Value >
          </ActionResult>
    </ActionResults >
   </Action>
</Actions>
<ActionTriggers>
    <ActionTrigger>
    <Condition>($status$=="ERROR")&&($notified$=="NO")
    </Condition>
    <Triggers>
          <Trigger>
                <TYPE>ACTION</TYPE>
                <ID>urn:sid:fr:samsung:TechnicalSupport.
                    AirCondition_X100</ID>
                <Security>("tech_Login","psw")</Security >
                <Actions>
                  <Action>
                    <ActionName>NOTIFY</ActionName>
                    <LaunchTime> </LaunchTime >
                    <Parameters>
                        <Parameter>
                           <ParaName>p1</ParaName>
                           <Value>$tech_email_addr$</Value>
                        </Parameter>
                        <Parameter>
                           <ParaName>p2</ParaName>
                           <Value>$AlertEmai$</Value >
                        </Parameter>
                    </Parameters >
                  </Action>
                </Actions>
          </Trigger>
    </Triggers >
   </ActionTrigger>
</ActionTriggers >
```

Figure 4. UODL format example

References

[1] ITU Internet Reports 2005: The Internet of Things.

[2] T. Laurent. "What's the deal with M2M or M2M2H?" This article is property of IQEnergy. 2009-2010. http://www.iqenergy.net/download/M2M_M2M2H_03232010.pdf

[3] European Technology Platform on Smart Systems Integration (EPoSS), "Internet of Things in 2020: Roadmap for the future". 2008, Version 1.1.

[4] J. Buckley, From RFID to The Internet of Things - Pervasive networked systems. Networks and Communication Technologies Directorate. CCAB, Brussels, 2006.

[5] L. Tan, N. Wang. Future Internet: The Internet of Things. 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE). 2010, Chengdu, China.

[6] N. R. Prasad, M. Eisenhauer, M. Ahlsén, Atta Badii, Andr?Brinkmann, Klaus Marius Hansen, Peter Rosengre. Vision and Challenges for Realising the Internet of Things. European Commission - Information Society and Media DG.

[7] L. Nagel, M. Roidl, G. Follert. The Internet of Things: On Standardisation in the Domain of Intralogistics. IOT 2008 WORKSHOPS, First International Conference on The Internet of Things. 2008, Zurich, Switzerland.

[8] J. Sen. Internet of Things - A Standardization Perspective. This article is property of Tata Consultancy Services. 2010

[9] S. Dominikus, M. Aigner, S. Kraxberger, Passive RFID Technology for the Internet of Things. The 5th International Conference for Internet Technology and Secured Transactions (ICITST-2010). 2010, London, UK

[10] http://www.w3.org/XML/

[11] R. Moats. "URN Syntax", RFC 2141, 1997.

[12] EPCglobal (2009), "The Application Level Events (ALE) Specification", Version 1.1.1, Part I: Core Specification. Section 6.2

[13] EPCglobal (2008), "EPCglobal Object Name Service (ONS)", Version 1.0.1.

[14] EPCglobal Standards Overview. http://www.gs1.org/gsmp/kc/epcglobal

[15] P. Mockapetris. RFC 1035, Domain names - implementation and specification.

[16] Java Remote Method Invocation RMI - Whitepaper. This article is property of Oracle.

[17] T. Dierks, E. Rescorla. The Transport Layer Security (TLS) Protocol. RFC 5246 (SSL is the predecessor to TLS)

[18] P. Srisuresh, K. Egevang. RFC 3022, Traditional IP Network Address Translator

[19] P. Srisuresh, M. Holdrege. RFC 2663, IP Network Address Translator (NAT) Terminology and Considerations

[20] M. Wu, T.Lu, F. Ling, L. Sun, H. Du. Research on the architecture of Internet of things. 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). 2010. Chengdu, China

[21] Y. Huang, G. Li. Descriptive Models for Internet of Things. International Conference on Intelligent Control and Information Processing. 2010. Dalian, China

[22] EPC (Electronic product code): from Wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/wiki/Electronic_Product_Code

[23] N. Kong, X. Li, B. Yan. A Model Supporting Any Product Code Standard for the Resource Addressing in the Internet of Things. First International Conference on Intelligent Networks and Intelligent Systems (ICINIS). Wuhan, China.

[24] B. Khoo. RFID - from Tracking to the Internet of Things: A Review of Developments. IEEE/ACM International Conference on Cyber, Physical and Social Computing. 2010. hangzhou, China.

[25] B. Xu, Y. Liu, X. He, Y. Tao. On the Architecture and Address Mapping Mechanism of IoT. International Conference on Intelligent Systems and Knowledge Engineering (ISKE2010). 2010. Hangzhou, China.

[26] F. Thiesse, c. Floerkemeier, M. Harrison, F. Michahelles, C. Roduner. Technology, Standards, and Real-World Deployments of the EPC Network. Page36-43, Vol 13.2. IEEE Internet Computing Journal.

[27] Rajan M.A, P. Balamuralidhar, Chethan.K.P , Swarnahpriyaah.M. A Self-Reconfigurable Sensor Network Management System for Internet of Things Paradigm. 2011 International Conference on Devices and Communications (ICDeCom). 2011. Ranchi, India.

[28] Q. Zhu, R. Wang, Q. Chen, Y. Liu, W. Qin, "IOT Gateway: BridgingWireless Sensor Networks into Internet of Things," euc, pp.347-352, IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010, Hong Kong, China.

[29] Y. Huang, G. Li. A Semantic Analysis for Internet of Things. ICICTA 2010 : International Conference on Intelligent Computation Technology and Automation. 2010. Changsha, China.

[30] I. Grønbæk. Architecture for the Internet of Things (IoT): API and interconnect. The Second International Conference on Sensor Technologies and Applications. 2008. Cap Esterel, France

[31] L. Tan, N. Wang. Future Internet: The Internet of Things. 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE). Chengdu, China.

[32] M. Yun, B. Yuxin. Research on the Architecture and Key Technology of Internet of Things (loT) Applied on Smart Grid. 2011 International Conference on Advances in Energy Engineering. Changsha, China, 2011

[33] H. Chaouchi. The Internet of Things: Connecting Objects to the web. 2010, ISTE Ltd and John Wiley & Sons Inc.