A Scalable Distributed Architecture Towards Unifying IoT Applications

Chayan Sarkar*, Akshay Uttama Nambi S. N.*, R. Venkatesha Prasad*, Abdur Rahim†
*Delft University of Technology, The Netherlands
†CREATE-NET, Trento, Italy
{c.sarkar, akshay.narashiman, r.r.venkateshaprasad}@tudelft.nl, abdur.rahim@create-net.org

***Abstract*—The advent of Internet of Things (IoT) has kindled the possibility of umpteen number of applications. One of the major challenges in the realization of IoT applications is interoperability among various IoT entities. Thus, the need for a new architecture – comprising of smart control and actuation – has been identified by many researchers. In this article, we propose a distributed, interoperable architecture for IoT, which will overcome most of the obstacles in the process of large scale expansion of IoT. It specifically addresses heterogeneity of IoT devices, and enables seamless addition of new devices across applications. We propose a layered architecture that provides various levels of abstraction to tackle the issues such as, scalability, heterogeneity and interoperability. Using a comprehensive study of a use-cases, comprising elements from multiple-application domains, we illustrate the usability of the proposed architecture.**

I. Introduction

In today's connected world, there are several means of ephemeral communication amongst devices, e.g., Bluetooth, WiFi, ZigBee, GSM, etc. However, the idea now is not only to connect with other communicating devices opportunistically, but also to be aware of various real-world non-communicable objects in the surroundings. This paradigmatic shift opens up new futuristic services. An important aspect of these services can be captured by the words of Mark Weiser. In his seminal paper [23], he provided a vision – "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it". This vision, in fact, is the driving force behind today's miniaturized technologies and communication substrates. Thus, we are about to witness a future where there will be thousands of inanimate objects for each person that will ceaselessly communicate with each other to support everyday life in a smart way. In general, this paradigm is referred to as the "Internet of Things" (IoT). The idea is to form an intelligent network of these humongous numbers of devices, systems and equipments. Further, the ambit of IoT is expanding to include any-'thing' that could be represented or identified in the cyber (virtual) world even without having any direct communication interfaces on those "things".

IoT can be viewed as a global infrastructure substrate that helps to connect objects (indeed any object) by exploiting the captured data and their communication capabilities, if any. All existing and evolving networks are presumed to be part of this IoT infrastructure. IoT will offer unique identification of the objects and their virtual representation as the basis for independent development of services and applications. These will be characterized by voluminous and self-governing data capture, event transfer, network connectivity and interoperability. In the following we enlist some of the important characteristics.

A. Desired Characteristics

Many IoT applications have been identified, e.g., smart home, smart logistics, smart transportation, smart health care, smart agriculture, etc. [7]. A common factor in all such applications is the inherent *smartness*. Being part of a "smart" application, various devices within an application domain can automatically collect data, share information among themselves, and initiate and execute services with minimal human intervention. Some of the desired characteristics of IoT objects (devices)[1] as well as IoT applications are listed below [15].

Automation: Automation is a key feature of any IoT device and application. Autonomous data collection and processing, making contextual inference, collaborating with other IoT objects and taking steps based on the current real-time circumstances should be supported by any IoT infrastructure.

Intelligence: Various objects in IoT should behave intelligently. Building intelligence in these objects to power them to operate adaptively based on different situations is an important aspect. Situation awareness is the key for an intelligent system, which can operate with minimal human intervention.

Dynamicity: An object in a IoT system or application can move from one place to another place, or one application domain to another. The IoT system should dynamically identify this and act accordingly.

Zero-configurations: To support easy integration of devices in the IoT ecosystem, plug-and-play feature should be available. Zero-configuration support encourages an easy and decentralized growth of IoT systems[22].

Having listed the characteristics of IoT devices and applications, we now summarize the technical challenges.

B. Technical Challenges

The main challenge is to manage and maintain large number of devices and react smartly according to the data generated by them. Some answers addressing this challenge can be seen under the umbrella of Future Internet and in projects such as BUTLER [1], COMPOSE [2], FIND [3], FIRE [4], IoT-A [6], etc. They deal with large scale networking, cognitive networking, network of networks[13], as well as service-oriented architecture development for a converged

[1]We use the terms IoT object and IoT device interchangeably.

communication and service infrastructure, to mention a few. In the light of growing IoT infrastructure, more number of sensors and actuators make the system more intelligent and highly responsive. Higher amount of sensed information and precise control could help in achieving sophistication. Furthermore, there must be many devices (to substitute) to make services fault tolerant and dependable. However, *paripassu*, they also impart difficulties of maintaining them, controlling them and filtering enormous amount of data generated by them.

In this light, we enlist some of the key factors that dictate the challenges in IoT related research.

Heterogeneity: IoT Devices are deployed by different persons/authorities/entities. These devices have different operating conditions, functionalities, resolutions, etc. Thus, enabling seamless integration of these devices is a huge challenge. The degree of complexity increases many folds when some of these simple devices are merged to form a complex network.

Scalability: The rapid growth of embedded technologies is leading to enormous deployment of miniaturized devices (sensors, actuators, etc.). As the number of devices grows, the data produced by these devices grow unboundedly. Thus, handling the growth of number of devices and information they produce is a massive challenge in IoT.

Interoperability: In an IoT application, there are many actors comprising of human and non-human objects. An actor can play multiple roles based on the current situations and environment such as, available resources in the IoT application, data provider, data consumer, service provider, etc. Seamless interaction amongst the various actors is crucial to envisage the vision of IoT. The interaction amongst different objects magnifies, especially when each actor is managed differently.

Security & Privacy: Due to the large number of actors involved in IoT, ensuring data authentication, data access control, data consistency and protection of data are a few core issues. To evolve a holistic system design, information security, privacy and data protection need to be addressed properly.

C. Our Contribution

In this article, we propose a layered and distributed architecture for IoT, called *Distributed Internet-like Architecture for Things (DIAT)*. We believe that it has a potential to tackle many of the technical challenges described earlier. It also supports the desired characteristics of IoT objects and applications. Our key contributions are listed below:

- We define a layered IoT architecture. Layering binds similar functionalities together and provides a hierarchical structure for the functionalities. It also provides decoupling of orthogonal features and encourages their independent development.
- Using a detailed description of our layered architecture, we show how heterogeneity, scalability and interoperability can be tackled.
- We ensure integration of the desired characteristics of IoT systems with the help of various cognitive functionalities defined at various layers of the architecture.
- With the help of a composite (cross application domain) use-case, we showcase that the proposed architecture (DIAT) has *distributiveness*. It is also capable of tackling heterogeneity, scalability and interoperability and enriched the features like automation, intelligence, zero-configuration, etc.

The rest of the article is organized as follows. First, we briefly discuss some of the existing IoT architectural design principals in Section II. In Section III, we describe our proposal of a layered architecture for IoT in detail. In Section IV, we provide a use-case example to validate our proposal and compare its features with some of the other architectural proposals. Finally, we conclude the paper in Section V.

II. Related Work

Many proposals attempt to define an architectural model for IoT that are usually applicable to a specific application domain. For example, Castellani *et al.,* have proposed an architecture for a smart office application [9]. Their main focus is only to interconnect wireless sensor networks and actuator networks to the Internet as a web service. Similarly, the EPC global IoT architecture has mainly focused on the RFID network and smart logistics system [8]. There is no suitable unified architecture till date that is appropriate for a global IoT infrastructure. The existing architectural proposals are defined for a particular type of application.

Many researchers have suggested layered architectures for IoT. For example, Gronbaek *et al.*, [11] and Dai *et al.*, [10] have proposed architectural model similar to the OSI architecture. In their design, the OSI stack is mapped to IoT. Tan *et al.*, have also suggested a layered architecture for the IoT [21]. However, their design failed to emphasize how to tackle the key challenges in IoT such as, interoperability, scalability, etc. There are other works that provide directions for the development of IoT architectures. Ning *et al.*, have provided a comparative study between man-like neural system and social organization framework for IoT architecture development [16]. Kovatsch *et al.*, have proposed a centralized web-resource based architecture for decoupling the application development from the domain of heterogeneous device [14]. However, these approaches neither guarantee scalability nor tackle interoperability of objects belonging to various application domains.

From the IoT perspective, every object can be seen as a potential service provider. However, these tiny services might not provide a complete application/solution. IoT applications need to holistically combine such multiple tiny services to provide complete functionality. They add up in right measures and sequence to provide a complete solution to users. In Service-Oriented Architecture (SOA), computers run an arbitrary number of services, and services can exchange information among themselves without human intervention and without the need to make changes to the underlying program itself. Thus, SOA approaches are considered to be applicable for IoT by many researchers [12], [20]. IoT is considered as a new service enabler, therefore, SOA based approaches can lead to some significant solutions. However, the architecture needs to be adopted in such a way that the technical challenges related to IoT can be tackled.

Overall, there are many pieces of solutions to enable IoT for smart applications; but a comprehensive and holistic

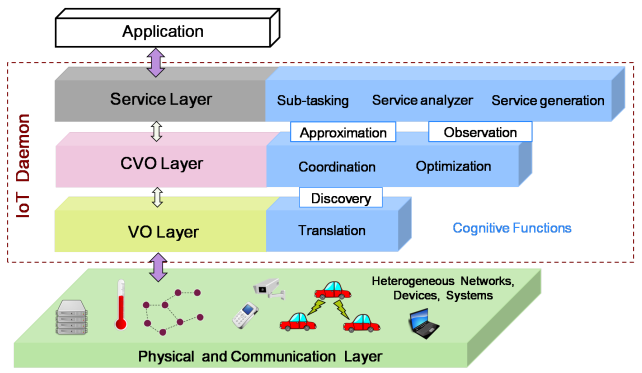

Fig. 1: A layered architecture for the Internet of Things.

design is required. In this paper, we attempt to provide such a comprehensive design.

III. IoT Distributed Architecture

Layering is a structuring technique that helps to group similar functionalities and encourages independent development of each layer. We follow a layered architecture for the IoT infrastructure. In IoT applications, similar to a service-oriented architecture, multiple services from individual service provider (or even individual objects) need to be merged with minimal human intervention. To fetch services from the objects without human interaction and to ensure easy collaboration among services, the objects need to expose their services in a similar way. Hence an abstract representation of the objects along with their features and capabilities is needed. On top of that, smart collaboration and coordination are required among the set of services that can serve towards a common goal. To create and manage the services, various service requests need to be properly analyzed before the execution. Thus, the functionalities of IoT infrastructure are grouped into three layers – (i) Virtual Object Layer (VOL), (ii) Composite Virtual Object Layer (CVOL), and (iii) Service Layer (SL). The three layers are responsible for object virtualization, service composition and execution, and service creation and management respectively.

To support some the key features of IoT, like automation, intelligence, etc., a set of cognitive functionalities is integrated at each layer of the architecture. The three layers of the IoT architecture and their main functionalities are put together as a stack, called *IoT Daemon* (Figure 1). The *IoT Daemon* forms the basis for the distributed architecture, which we call "**D**istributed **I**nternet-like **A**rchitecture for **T**hings (**DIAT**)". Please note that the lowest layer (VOL) is mainly responsible for virtual representation of objects and acts as an interpreter between the physical and the cyber worlds. Features such as management, coordination, automation, etc., are implemented in the upper layers. Therefore, the cognitive functions as well as the resource requirement (e.g. processing power, memory, etc.) increases from lower to higher layers. Here we do not consider building physical communication amongst the devices since it has been dealt in depth in the literature. In the next subsections, we describe the roles of each layer and some of their cognitive features, before describing the *IoT Daemon* in detail in Section III-E.

A. Virtual Object Layer

The *Virtual Object Layer (VOL)* is responsible for virtualization of physical objects or entities. It hosts the virtual representations of the objects, called *Virtual Object (VO)*. A VO provides a semantic description of the associated real-world (or physical) object, which follows a generic structure for all types of objects. Semantic modeling of VO enables efficient information exchange by expressing the information and its relationship among other VOs. Through semantic technologies, the VO layer provides universal methodologies to access all physical objects. Thus, the inherent heterogeneity in various devices, systems and networks is tackled by the notion of VO. Furthermore, the VOL plays the role of bridging the gap between the physical and the cyber world. As VOs are the digital representation of physical objects, the objects can be accessible in the cyber world through its corresponding VOs. Indeed, the VOs act as a translator between the digital and physical worlds. This abstraction helps to tackle the heterogeneity and ensures interoperability and reusability of objects.

B. Composite Virtual Object Layer

An individual VO represents its corresponding physical object and it is constrained by the capability of that object. In reality, multiple physical objects need to be engaged in order to accomplish a particular task. They need to communicate and coordinate among themselves to fulfill the goals. A *Composite Virtual Objects (CVO)* does this job. At runtime, a CVO is created by forming a mash-up of one/multiple VOs (and/or other existing CVOs) based on the necessity of a task. *Composite Virtual Objects (CVO)* are formed in CVO Layer (CVOL). During the execution, a CVO dictates how the individual entities (VOs and/or CVOs) in its mash-up should work. The CVOL plays the role of a coordinator. Additionally, the CVOL tries to optimize the operations among the items by doing smart scheduling.

One of the key features of the CVOL is to locate a suitable VO (or CVO) that can accomplish a subtask of a service request. Thus, a *discovery* mechanism is required. It may reside in both VOL and CVOL to discover suitable VOs and CVOs respectively. Semantic description of a VO (CVO) easily describes its features and capabilities, and helps in the discovery mechanism. If an exact match for VO (CVO) is not found according to the requirement an alternative and an approximate VO that can fulfill the requirement partially is selected. Similarly, an exact service match (as requested) might not be available always. A close alternative can serve the purpose. This paradigm is termed as *approximate service* [19], [18], [17]. Similar to the discovery mechanism, the approximation capability is also spread across two layers; but in this case, the layers are CVOL and SL.

C. Service Layer

The *Service Layer (SL)* is responsible for creation and management of services. On one hand, it handles various service requests from users. On the other hand, it initiates service requests on its own in order to enable automatic service creation. Additionally, whenever a service request is received, the service layer analyzes and splits it into smaller

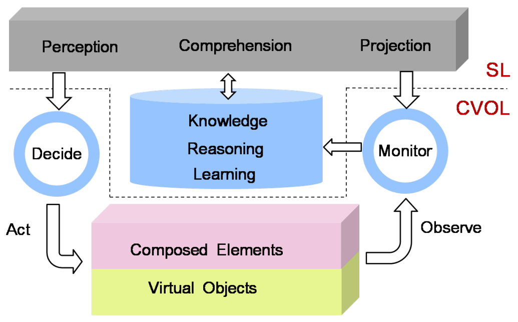

Fig. 2: Functionalities and work-flow of an observer.

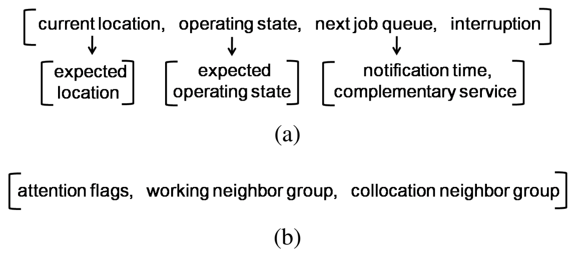
current location, operating state, next job queue, interruption
expected location
expected operating state
notification time, complementary service
(a)
attention flags, working neighbor group, collocation neighbor group
(b)

Fig. 3: Contextual information vector (a) for a human object; (b) for a non-human object.

subtasks. The service layer also decides how these subtasks are assembled to reach the final goal. This (abstract) description of a decomposed service request is provided to the CVOL to execute the task.

D. Observer

One of the major tasks of the service layer (SL) is automatic creation of services according to the situations. The *observer* plays a key role in the automation of machine-to-machine (M2M) communication in order to provide a service intelligently. The functionalities and the work-flow of an *observer* are spread across the CVOL and SL (Figure 2). The observer continuously monitors objects and assesses their situation. Based on available knowledge and current situation, it may decide to initiate a service request and some necessary matching service(s).

Ubiquitous and pervasive computing is an inherent feature of IoT. Thus, context-awareness is an essential requirement for IoT. To initiate and execute an automated service intelligently, situations of all objects (humans, devices) need to be assessed carefully. Hence, context-awareness is an integrated part of the *observer*, which continuously monitors objects and derives the contextual information. The *observer* stores the contextual information about each object in its associated vector. Objects are broadly categorized into two groups - human and non-human objects. Two different types of vectors are used for these two sets of objects. A contextual information vector for a human object contains the following fields (refer to Figure 3a).

1) Current location: This field contains the current location of the human being. This is not raw data (like GPS data), but some high-level information derived using low level data. For example, the current location can indicate *atHome, atOffice, atSupermarket*, etc. Additionally, this field has a sub-field, called *expected location*, which is derived based on the pre-scheduled job list and previous observations. If expected location information is available and if it differs from the current location, the user will be notified or a new service request will be initiated. For example, if the current location is *atHospital* and expected location is *atOffice* due to a scheduled meeting, a service request to generated to inform all the interested party for cancellation and rescheduling of the meeting.

2) Operating state: The operating state indicates what a person is currently involved in. Examples of operating state can be *inMeeting, isWorking, watchingTV, isSleeping*, etc. This helps to initiate new service requests. For example, if a person is in a meeting, any incoming call is automatically diverted to auto-response mechanism or another person and the actual callee is notified in due time. Similar to the location field, this field also has a sub-filed, called *expected operating state*. Any conflict between the sub-field and the main field initiates a new service request.

3) Next job queue: This field describes upcoming jobs that are derived from the to-do list or calendar events of a person. This field contains two sub-fields - *notification time* and *complementary service*. The notification time is the approximate time when an event should happen. The SL initiates a new service request at that moment. Additionally, the execution of a job might initiate some other jobs. The *observer* tries to determine those jobs and queue them in the list. The complementary service field points those jobs in the queue.

4) Interruption: Any service request (human-initiated or auto-generated) might require some human intervention at various points during the execution. The *observer* sets an interruption flag, if some service requires human attention. Based on the urgency level, a person is either notified immediately or it is postponed. The operating state of the person is important to decide the urgency level. For example, if a person is in a meeting and the central heating system of his/her house is broken, the person can be informed later. But if there is a burglary, the person should be informed immediately.

Similarly, the contextual information vector for a non-human object contains the fields as shown in Figure 3b.

1) Attention flags: This field contains various flags that indicate something is unusual with the object and attention is needed. The attention may not be necessarily from human beings. It may be in consultation (communication) with other objects. The level of urgency decides how quickly the attention needs to be provided. For example, an *observer* identifies a fire from the data produced by the fire detector VO and sets the attention flag to a value so that an immediate fire alarm service request can be initiated. On the contrary, if a room heater stops working, the attention flag is set to a relatively low priority value such that a service request is scheduled at an appropriate time.

2) Working neighbor group: This field creates a group amongst similar objects within a small area. The definition of an area can vary in different scenarios. In case of climate controller in an office domain, the area is defined by the office premises (building). On the other hand, the area for a traffic sensor can spread across couple of blocks. The group members can communicate with each other for coordination and optimize their performance. An Observer can detect an anomaly in one object by consulting with other group members and set the attention flag.

3) Collocation neighbor group: Similar to the working neighbor group, this field creates and monitors a group among the neighboring objects. These objects need not be of the same type, but they are grouped based on their geographical collocation.

Next we describe the concept of *IoT daemon* in detail. The *IoT daemon* is executed in each device and it binds many objects for providing smart applications.

E. IoT Daemon

All the proposed layers and their functionalities together is referred to as an *IoT Daemon*. The IoT daemon acts as the basis of the distributed architecture in DIAT. The abstracted view of an IoT daemon is shown in Figure 1. Every object with some processing power and memory runs its own IoT daemon. The footprint of the daemon varies with the capability of the devices and available resources. The presence of the IoT daemon converts an ordinary device into an IoT device or makes it IoT compatible.

A full-fledged *IoT daemon* contains all the three layers with all the functionalities. But some tiny embedded devices might lag processing power and memory to run a full version of the daemon, e.g., wireless sensors, actuators, etc. A depreciated *IoT daemon* may run on those devices with limited set of functionalities. For example, a basic VOL with the capability of hosting at least one VO is executed in a wireless sensor node. This implies that no centralized hosting of VO is required.

The presence of IoT daemon in every object can relieve the burden of configuring each device manually, i.e., ensuring the scalability and zero-configuration requirement[22]. The VOs in the VOL represents each object in digital/cyber domain. This representation abstracts the heterogeneity of devices in terms of capabilities and features. So, the VOL tackles the heterogeneity whereas the APIs and interconnections among the three layers ensure interoperability. The automatic service creation, smart execution, and dynamic adjustment are provided by the CVOL and SL. The cognitive capabilities of the *observer* ensures that services can be initiated and executed automatically, i.e., it ensures maximum End-to-End communication with minimal human intervention.

F. Comparison

We compare our proposal (DIAT) with some of the existing and ongoing efforts based on potentiality of tackling technical challenges and availability of the key characteristics of IoT (Table I). From the table of comparison, it is evident that our proposal shows a great promise for a global IoT architecture.

IV. A MULTI-APPLICATION USE-CASE

To validate our architecture, we describe a composite use-case in this section. The example consists of various IoT applications like smart home, smart transportation, smart health-care, etc., as shown in the Figure 4. The proposed architecture

TABLE I: Comparison among Architectural Proposals.

IoT Characteristics	DIAT	Butler [1]	Compose [2]	IoT-A [6]
Heterogeneity	yes	partially	yes	yes
Scalability	yes	no	no	–
Interoperability	yes	no	partially	yes
Automation	yes	yes	partially	yes
Zero-configuration	yes	no	no	partially
Distributiveness	yes	partially	no	–
Layered design	yes	no	yes	no

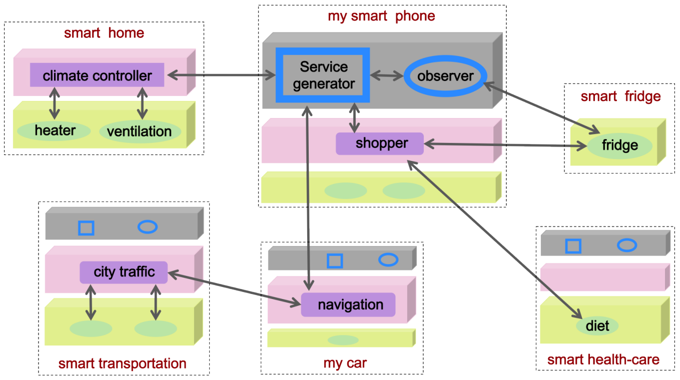

Fig. 4: A use-case showing the unification of various applications based on the distributed IoT architecture.

overcomes the barrier of (so called) separate IoT applications and achieves a global *Internet of Things* via interoperability. We assume an IoT daemon runs on every object, belonging to various application domains, and they communicate amongst themselves to carry out a task.

In the example shown in Figure 4, a smart fridge hosts a simple IoT daemon, which has only the VOL. The IoT daemon hosts a *fridge* VO, and it can inform a legitimate user (e.g. the owner) about its content. An *observer* of the fridge is hosted in a more capable pairing device, e.g., a PC, a smart phone, etc. The *observer* can identify essential items that are not available in the fridge. Let us assume that the *observer* is in the smart-phone of owner of the fridge and it may add a task in the "next job queue" if some (essential) items are missing from the fridge. When the owner leaves his/her office in the evening, the *observer* identifies this by analyzing the contextual information vector associated with the person (current location and operating state). It also fetches the scheduled tasks from the job queue and creates a service request through the service generator. The service request forms a CVO, called *shopper*, by engaging two VOs – the *fridge* VO and the *diet chart* VO. The *diet chart* VO is maintained at the "smart health care" domain. It is created based on doctor's and/or dietitian's advice. The *shopper* CVO also finds a convenient place (supermarket) to buy the items. From the contextual information vector, the *observer* also identifies that the person is driving currently. So the *service generator* informs the *navigation* CVO, residing in the car's IoT daemon, to fetch the route information to the nearest (or on the way to home) store. Then the *navigation CVO* gathers the information about the shortest route and an empty parking place from the *city traffic* CVO of the "smart transportation system". The *city traffic* CVO collects these information after consulting the appropriate VOs. Finally, the store is selected based on traffic condition, parking information and distance.

Additionally, the *service generator* estimates the time to reach home. It conveys this information to the "smart home" climate control application. The *climate controller* CVO then engages the *heater* and the *ventilation* VOs to make the indoor climate as comfortable as possible according the preference of the owner.

V. Conclusion and Future Work

The foundations for a generic IoT architecture have been discussed in this paper. Based on these foundations, we have proposed a distributed layered architecture for the IoT, called DIAT. DIAT is simple, scalable, accommodates heterogeneous objects, and also allows interoperability among these heterogeneous objects. Features such as automation, intelligence, and zero-configuration are integral part of DIAT. We have shown that DIAT satisfies the key characteristics and goals of an IoT architecture through the study of a realistic use-case.

The basic concepts of a futuristic architecture is being shaped in an on-going EU project, called iCore [5]. A full-fledged architecture and implementation of couple of use-cases will be delivered at the end of this project through this architecture. We are incorporating security, privacy measures, and also use human inputs without explicit participation to achieve approximate services.

Acknowledgment

The research leading to these results has been performed within the iCore project (http://www.iot-icore.eu/) and received funding from the European Commission's Seventh Framework Programme (EU FP7) under grant agreement no. 287708. We also thank Go-Green project (http://gogreen-project.nl/). Go-Green is sponsored by the Dutch Ministry of Economic Affairs, Agriculture and Innovation.

References

[1] BUTLER. http://www.iot-butler.eu/. [Online].

[2] COMPOSE. http://www.compose-project.eu. [Online].

[3] FIND. http://www.nets-find.net/. [Online].

[4] FIRE. http://www.ict-fire.eu/. [Online].

[5] iCore: Empowering IoT Through Cognitive Technologies. http://www.iot-icore.eu/. [Online].

[6] IoT-A. http://www.iot-a.eu/. [Online].

[7] L. Atzori, A. Iera, and G. Morabito. Siot: Giving a social structure to the internet of things. *Communications Letters, IEEE*, 15(11):1193–1195, 2011.

[8] S. Beier, T. Grandison, K. Kailing, and R. Rantzau. Discovery services-enabling rfid traceability in epcglobal networks. In *COMAD*, pages 214–217, 2006.

[9] A. P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby, and M. Zorzi. Architecture and protocols for the internet of things: A case study. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 678–683. IEEE, 2010.

[10] G. Dai and Y. Wang. Design on architecture of internet of things. In *Advances in Computer Science and Information Engineering*, pages 1–7. Springer, 2012.

[11] I. Gronbaek. Architecture for the internet of things (iot): Api and interconnect. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, pages 802–807. IEEE, 2008.

[12] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio. Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. *Services Computing, IEEE Transactions on*, 3(3):223–235, 2010.

[13] J. Hoebeke, G. Holderbeke, I. Moerman, M. Jacobsson, V. Prasad, N. C. WANGI, I. Niemegeers, and S. H. De Groot. Personal network federations. 2006.

[14] M. Kovatsch, S. Mayer, and B. Ostermaier. Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 751–756. IEEE, 2012.

[15] G. M. Lee and J. Y. Kim. The internet of things: A problem statement. In *Information and Communication Technology Convergence (ICTC), 2010 International Conference on*, pages 517–518. IEEE, 2010.

[16] H. Ning and Z. Wang. Future internet of things architecture: like mankind neural system or social organization framework? *Communications Letters, IEEE*, 15(4):461–463, 2011.

[17] R. V. Prasad, E. Onur, V. S. Rao, Y. Durmus, A. R. Biswas, and I. Niemegeers. Approximate service provisioning in an invisible network of the future. *27th WWRF Meeting, Dusseldorf*, 2011.

[18] R. V. Prasad, C. Sarkar, V. S. Rao, A. R. Biswas, and I. Niemegeers. Opportunistic service provisioning in the future internet using cognitive service approximation. *28th WWRF Meeting, Athens, Greece*, 2012.

[19] C. Sarkar, V. S. Rao, R. V. Prasad, A. Rahim, and I. Niemegeers. A distributed model for approximate service provisioning in internet of things. In *Proceedings of the 2012 international workshop on Self-aware internet of things*, pages 31–36. ACM, 2012.

[20] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza, and V. Trifa. Soa-based integration of the internet of things in enterprise services. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 968–975. IEEE, 2009.

[21] L. Tan and N. Wang. Future internet: The internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–376. IEEE, 2010.

[22] N. I. C. Wangi, R. V. Prasad, M. Jacobsson, and I. Niemegeers. Address autoconfiguration in wireless ad hoc networks: Protocols and techniques. *Wireless Communications, IEEE*, 15(1):70–80, 2008.

[23] M. Weiser. The computer for the twenty-first century (pp. 94-100). *Scientific American, September Issue*, 1991.