Taking the SIoT down from the Cloud: Integrating the Social Internet of Things in the INPUT Architecture

I. Farris [a], R. Girau [b], M. Nitti [b], L. Atzori [b], R. Bruschi [d], A Iera[a], G. Morabito [c]

[a] CNIT – University Mediterranea of Reggio Calabria, {ivan.farris, antonio.iera}@unirc.it

[b] CNIT – University of Cagliari, {michele.nitti, roberto.girau, l.atzori}@diee.unica.it

[c] CNIT – University of Catania, giacomo.morabito@dieei.unict.it

[d] CNIT – Research Unit of Genoa, roberto.bruschi@cnit.it

***Abstract*—The Social Internet of Things (SIoT) is a paradigm which is rapidly gaining ground in the Internet of Things (IoT) domain. In the SIoT objects can establish social-like relationships between each others autonomously. In this paper a solution is presented that integrates the SIoT concept in the architecture proposed within the INPUT project. More specifically the feature is exploited of the INPUT project which allows for running the virtual representation of a smart/social object in the access router which is nearest to the physical object. In this way it is expected that delay will decrease and efficiency in the usage of network resources will increase.**

***Index Terms*—Social Internet of Things, network virtualization, cloud computing.**

I. Introduction

The effective deployment of large-scale IoT solutions is extremely challenging and requires supporting essential features such as dynamic composition of dependable services [1], efficient service discovery [2], and full interoperability among the plethora of involved IoT devices [3]. In this context, *Social Internet of Things* (SIoT) represents a promising paradigm allowing objects to establish social-like relationships between each other [4], [5]. The resulting network has several interesting features. In fact, it is well-known that social networks are structurally navigable, which is a fundamental feature to perform search of specific services in effective and efficient ways. Furthermore, by leveraging on social relationships it is possible for objects to rely on differentiated levels of trustworthiness, which might be useful to support security. Finally, social relationships can be established between objects that use different technologies, which is important to support interactions between objects across different IoT platforms.

Accordingly, in the recent past a large number of research efforts have focused on the exploitation of SIoT concepts.

Usually, solutions for the SIoT envision a server in the cloud which manages and stores the social-like relationships between different objects. Sometimes, such a server is also responsible for storing updates sent by objects. This is, for example, the case of solutions based on the ThingSpeak platform.

Solutions based on a (logically) centralized server in the cloud have several advantages but suffer from a few major problems. In fact, objects might be far away from the data center hosting the cloud, which results in long delays and inefficiency in the use of communication resources.

In this paper we exploit the approach proposed by the INPUT project to bring the virtual representation of objects closer to their physical counterpart. The INPUT project builds on Software Defined Networking (SDN) and virtualization techniques to efficiently provide next-generation cloud services with strict latency requirements. Moreover, to improve the quality of service, network infrastructure can only exploit part of users' acceptable latency times due to the overall computational burden of relevant applications, as demonstrated by a recent study of Nokia [6] (see Fig. 1).

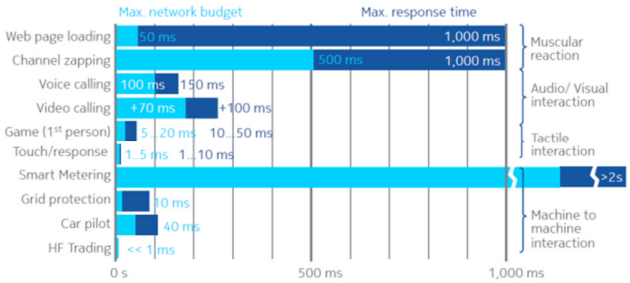

Fig. 1. Maximum latency network budget for real-time cloud services [6].

Furthermore, moving the virtual representation of objects closer to their physical counterpart can increase the efficiency in the use of resources. In fact, in the INPUT architecture, access routers of Internet service providers are equipped with processing and storage capabilities that can be used to run virtual machines. Such virtual machines execute user application functions to reduce the latency and improve resource and energy efficiency.

In our scenario the access router will provide PaaS services, where virtual representations of the objects can run. An Orchestrator will decide which virtual representation of the objects is most convenient to move close to the physical device. Such decision is a function of the rate with which interactions between pairs of (or several) nodes occur. An SDN Controller will then update routing in such a way that the virtual representation of an object can be reached in its new location.

The rest of this paper is organized as follows. In Section II we provide an overview of the relevant literature and research

activities. In Section III we first present the INPUT architecture and then describe how the SIoT can take advantage of it. In Section IV we describe the implementation of the proposed solution. Finally, in Section V we draw our conclusions.

II. Related work

Objects in the Internet of Things (IoT) will act in a very complex environment characterized by a large number of difficulties and threats. One of the most promising approaches proposed to face such a complexity is to provide objects with *social*-like behavior [5]. The resulting concept, named *Social Internet of Things* (SIoT), has been formalized and analyzed in [4].

Realization of the SIoT requires a question to be addressed: *who should manage social relationships between objects and where should the relevant information be stored and processed*?

In most cases objects in the IoT are resource constrained and therefore, it is unlikely that they can handle *sociality* procedures by themselves. Indeed, to the best of our knowledge there are no solutions in the literature that propose local (inside the devices themselves) processing and storage of social relationships.

There are, instead, solutions that envision interactions between social objects to occur thorugh traditional and/or customized social network websites. Examplary solutions in which interactions occur through traditional social networking websites such as Facebook and Twitter include Toyota Friend Network [7], Nike+ [8], and Lifely [9]. Other solutions exist in which social interactions between objects (and people) occur through *ad hoc* social networking websites. Examples of this class of platforms include the Social Web of Things developed by Ericsson [10]. Major advantage of such solutions is that they build on interaction modes which are familiar to users; however, they do not specify how relationships between objects and/or between objects and people should be created and which policies should be applied for their management. In other words they leave the management of relationships application dependent. The consequence of such approach is that relationships established between objects in the context of an application cannot be used by other applications.

An alternative approach is to deploy a server which gather information about objects status and activities in order to discover whether it is the case for a pair of objects to establish a social-like relationship. Such a server will offer an API to application developers who can use it to implement new applications. An example of this type of solution is proposed in [11]. Usually the above server is deployed in the cloud. Such a deployment choice has several advantages typical of cloud-based solution, but lacks in efficiency and timeliness. In fact, in most cases the data centers offering cloud services are located far from the objects, which results in high communication delays and scarce efficiency in the usage of networking resources.

In this paper, we analyse how to overcome the above problems by adopting the approach proposed in the INPUT project. In fact, as we explain in the following Section III, in the INPUT concept, part of the cloud service is moved to the access router of the network provider which is closest to the objects.

III. The INPUT Platform

The European H2020 INPUT (In-Network Programmability for next-generation personal cloUd service supporT) Project [12] aims at exploiting SDN and NFV (Network Function Virtualization) [13] paradigms to achieve breakthrough towards a fully "softwarized" network of Telco providers, enabling next generation cloud applications. In particular, the objective is to (i) dynamically deploy application-level services and network-specific functions in the edge network devices, (ii) design a distributed architecture composed by computing and storage appliances and by physical/virtual SDN switches, and (iii) move cloud services closer to end-users.

A. INPUT architecture

Virtualization represents a fundamental aspect of the INPUT architecture, which aims to replace physical Smart Device (SD) with their "*virtual images*", thus introducing the concept of *Smart-Device-as-a-Service* (SDaaS). This novel paradigm can be exploited:

- to fully dematerialize classic user's home appliances (e.g., set-top-boxes, video recorders, network-attached storage server, etc.) and provide all their functionalities by the cloud;
- to add potentially infinite smartness and capacity to resource-constrained IoT devices (such as sensors and actuators), whose physical counterparts must be maintained to perform their location-based monitoring and control functions.

Virtual and physical SDs will be made available to users at any time and at any place by means of virtual cloud-powered *Personal Networks* (PNs), which will constitute an underlying secure and trusted communication service. These PNs will provide users with the perception of always being within their home Local Area Network composed of their own (virtual and physical) SDs, *independently from their location* (see Fig. 2).

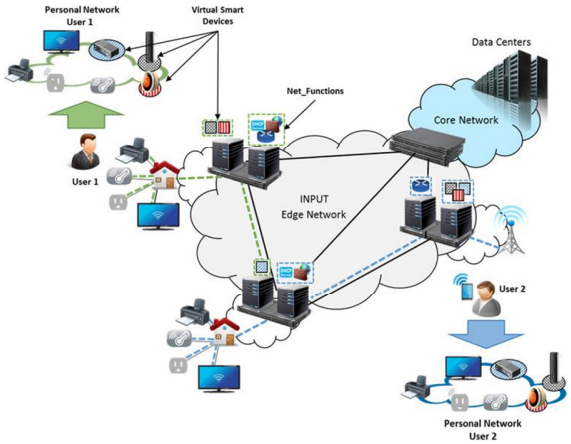

Fig. 2. Mapping of the Users' Personal Network in the INPUT edge network [12].

To achieve these ultimate objectives, the INPUT Project will enhance edge network devices with computing and storage capabilities, enabling the so-called "in-network" programmability. This will allow edge network devices to host cloud service applications, which will cooperate with running services in users' terminals and remote datacenters.

On the other hand, physical SDs typically connected to the user's LAN (e.g., network-attached storage servers, set-top-boxes, video recorders, home automation control units, smart meters, etc.) will be fully or partially virtualized in the INPUT computing facilities by software instances, named Service Applications (*Service_App*), running at different levels of the edge network infrastructure and providing application layer services.

B. Benefits of INPUT platforms for IoT applications

The INPUT platform provides the ideal substrate to deploy a win-win solution for service cloud providers, Telco provider and users. The current landscape of IoT cloud platforms foresees the virtual representation of IoT nodes. This could imply lack of interoperability among the interfaces defined for the virtual objects and cause significant latency due to the remote location of datacenters. IoT applications present extremely different QoE (Quality of Experience) requirements, ranging from real-time services with stringent deadlines to multimedia applications with strict requirements in terms of bandwidth and jitter, to delay-tolerant scenarios where sensing information should be stored in remote servers, introducing big data issues for their storage and processing. Therefore, the INPUT platforms allows service cloud developers to deploy new cloud-based IoT applications, able to guarantee those service requirements which could not be satisfied by remote datacenters (far from the user). Furthermore, the inherent distributed architecture of INPUT enables scalability features, avoiding server overloading by distributing the load among multiple micro datacenters in the access points of the network.

The Telco providers could implement SDN and NFV paradigms to innovate their network and to open new business areas, offering cloud services from a privileged position. In fact, NFV allows to deploy on-demand services in the telco nodes closer to the users, matching the manifold requirements of applications and allowing for a reduction of Operating and Capital Expenses (OPEX and CAPEX). Thus, Telco providers could increase their revenue opportunities in the promising IoT domain, offering not only network services but also shifting their role to a higher position in the value chain.

These advantages have to be analyzed considering that whereas the provisioning of computing and storage resources are typically location-independent, sensing activities are strictly correlated to their physical positions. According, with a centralized cloud platform the data should be sent from the sensing device position to the central servers and then back to the user. With the distributed cloud solution, sensing information could be processed closer to the devices, thus reducing traffic congestion in Telco network. Furthermore, more processing power close to the user can extend the number of applications and achieve a better QoE experience.

Finally, as remarkable energy consumption reduction is envisioned by the INPUT platform, the advanced virtualization technologies will allow to partially dematerialize end-user IoT devices (zeroing their embodied carbon footprint) and to host their virtual images on a more energy-efficient network infrastructure.

IV. INTRODUCING THE SIOT CONCEPTS INTO THE INPUT PLATFORM

A. SIoT integration and added-value IoT applications

Virtualization of devices is a fundamental aspect in the INPUT platform. The SDaaS paradigm allows for creating virtual representations of user's devices. These virtual representations could implement all the functionalities of a device, which could even exist in the virtualization layer only, e.g., the virtual Smart Devices number 4 (vSD_4) in Fig. 3. For instance vSD_4 may be a process running in the cloud which collects physical world information from the web without having a reference physical Smart Device (pSD) counterpart. For devices like IoT nodes the virtual images allow for a standardized interface to access their heterogeneous resources and maintain up-to-date metadata, such as: object type, computational power and mobility capabilities. Finally, the service layer can aggregate and compose the different vSDs' functions to create advanced cloud service applications.

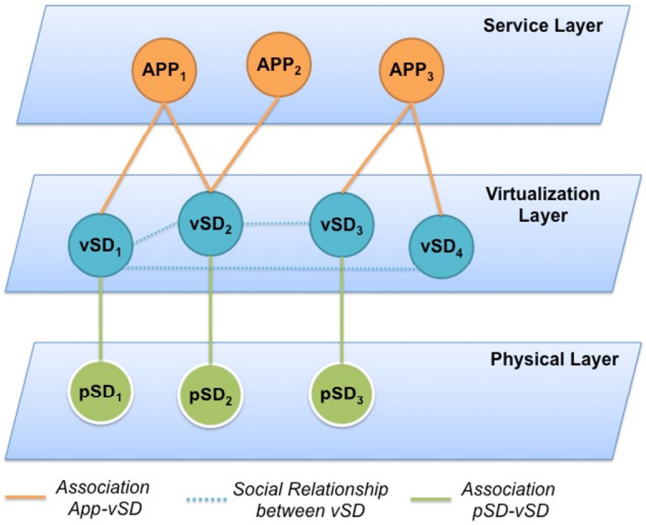

Fig. 3. Logical associations between physical and virtual Smart Devices

In this paper, without losing generality, we refer to the concept of the Social IoT model implemented in [11]. To integrate the SIoT concepts in the INPUT platform, a middleware is needed to enable the *"virtual image"* of the physical SD to perform social functionalities and then move the virtual representation of the physical SD from the cloud into the edge nodes of the INPUT platform.

In this platform, every "type" of physical object can be described by a Profile, which consists of a semantic description of the object and its capabilities, resources and permissions. The virtual representation of a SD is actually an instance of the

Profile and represents the actual web service running on the Cloud. Social functionalities are assured by the presence of a middleware, common to all the Profiles, which is needed in order to allow the virtual objects to exploit their own relationships.

Depending on its capabilities, the physical device is connected to its virtual counterparts through different communication models and procotols, from HTTP over SSL to MQTT. Additionally, to address the dissimilarities in the hardware characteristics of the devices an abstraction layer of the hardware is used. This is in charge of creating an encrypted secure point-to-point communication channel with the vSD. The association among the physical SD and the vSD is done during registration when the two astraction layer entitities on both cloud and device exchange object IDs, URL and encryption of the key.

vSDs can access resources based on different permissions: public, private and friend. Resources exposed as pubblic do not need an access key and are then available to anyone. A private permission means that only the owner of the object, and its applications, can access the resources, through the use of an Owner Key, while if the permission is set to "friend", then the resources are shared with other vSDs when a relationship is established, through the exchance of a Friend API Key.

The social middleware exposes the following functionalities. The *relationship management* introduces the intelligence that allows objects to start, update, and terminate relationships, on the basis of the objects features (such as: object type, computational power, mobility capabilities, brand) and activity (frequency in meeting the other objects). For example, some relationships are determined by the static characteristics of the object (or slowly varying characteristics): type, brand, ownership. The other kinds of relationship are determined by the movement of the object and by the other objects it comes across. Clearly, the configuration of these functions is controlled by the object owner; accordingly, the resulting links can be asymmetrical.

The second component is the *service discovery* that has the purpose of finding which objects can provide the required Service App in the same way humans seek for friendships and information, i.e., by navigating the social network of friends. When a user needs a particular cloud service, this module is activated in one of the vSDs of his/her personal network, e.g., in the vSD associated to the smartphone, to find the Service Apps required to perform the specific tasks. The process starts with the smartphone's vSD looking for friends in its local database, which can provide the requested Service Apps; if none of the friends is a positive match, then the request is forwarded to other friends by making use of social measures, such as centrality or local clustering, until a set of objects capable to provide the needed Service Apps is found. If, by any chance, the smartphone vSD is down, the system is able to start the discovery process from any other object inside the user personal network, thus avoiding single point of failure.

To construct the chain of Service Apps, a *service composition* module is required, which enables the interaction among vSDs. However, in the IoT scenario, many objects, and consequently vSDs, will be able to provide the required Service App(s), therefore to build a reliable system a *trustworthiness management* module [14] is needed as a mean to understand how the information provided by other friends has to be processed on the basis of their behavior. To this aim, since the forms of socialization among objects have been devised to represent the human relationships, they can be ranked based on the type of relation that links two objects. Between two vSDs that belong to the same owner, and therefore are in the same personal network, we can infer a higher level of strength with respect to the relation between two objects of the same brand but that potentially never met.

The SIoT solution seems to provide several promising added-value features for the INPUT architecture. Indeed, the SIoT structure can be shaped by creating or deleting relationships in order to allow vSDs to easily find the desidered Service Apps, for example by creating many connections for popular vSD. Moreover, a mechanism of resource discovery based on social relations is intrinsically distributed, since every vSD can search for information by crawling its own social network and no central database is needed. In the same way, it is possible to use the social network to define model for the management of the level of trustworthiness among the objects not reachable in other ways.

V. Challenges For Implementation

To enable the virtual objects to exploit social capabilities, we need to face the following challenges:

A. Virtualization platform

A Platform as a Service (PaaS) is required to deploy and run any virtualization in the Cloud. First, a container is needed. The choice of the container depends on the virtualization type: it is possible to use an active virtualization, like a web-service that has its own independent code, or a passive virtualization, like a representation on a database. The choice of a database server is another important choice to be done: relational or NoSQL database? The first is reliable in terms of consistence of transactions but suffers in cloud performance and it is stiffer due to the use of fixed structures (tables). The second is more suitable to a changing environment such as the IoT, but the fluidity of a dynamic cloud could affect consistence of data.

B. Naming Service

In order to reach the actual vSD running instance a naming service is needed. It must track any change of the cloud network as regards to the locations where vSD have migrated to. Any communication in the social network should be possible without the need of knowing where vSDs are actually running. A CaaS (Container as a Service) cannot provide a way to map static IP addresses to a vSD. In order to optimize the network path between an end user and a vSD (or between two vDSs), end users on different ISPs or geographic locations might use different IP addresses to access the same vSD. DNS might return different IP addresses to access vSD over time or from different network locations. A possible solution could be the use of a hierarchical DNS system combined with Persistent URLs (PURLs) Type 303, which is used for Web Resources.

When this solution is selected, the PURL mechanism will perform the dynamic monitoring of vSD addresses updating current temporal addresses and maintaining the persistent address for each vSD [15].

C. vSD migration handling

In order to manage vSD migrations from an edge node to another a controller/orchestrator is needed. It should be able to move a vSD code to a container and the related data to a database on another edge node. The migration must be accomplished in a seamless way to avoid service delivery failures. The migration decisional logic could be on the controller or on vSDs. In the first case, the controller has to observe vSD interactions and it has the algorithm needed to make a choice of migration. In the second case, vSDs decide when migrate and ask for a new deploy to the controller. Lastly, the choice of using a centralized controller (although in cloud) or a hierarchically distributed controller system depending on the cloud partition into *zones*.

D. Social network management

Social information on vSDs is crucial to fully exploit SIoT advantages and where it should be stored is to be taken carefully into account. In a subjective scenario, vSDs only have their own view of the social network. On the other hand, in an objective scenario, social information is kept on a level on top of the vSD level and is accessed by vSDs and other actors in the same way. Finally, a hybrid solution could give the best compromise when elements out the social network need this information (e.g. deploy controller).

VI. CONCLUSIONS

In this paper, we have discussed the advantages that the solution envisioned by the INPUT project can bring in the application of the SIoT concepts. In fact, we have discussed that by bringing the virtual images of the IoT nodes close to their physical counterpart it is possible to reduce delay and increase efficiency of network resources. Furthermore, we have presented how such integration can be achieved as well as the technical challenges which need to be addressed.

ACKNOWLEDGMENT

This work has been supported by the INPUT (In-Network Programmability for next-generation personal cloUd service supporT) project funded by the European Commission under the Horizon 2020 Programme (Call H2020-ICT-2014-1, Grant no. 644672).

REFERENCES

[1] L. Atzori, A. Iera, G. Morabito, "The internet of things: A survey," Computer networks 54.15 (2010): 2787-2805.

[2] A. J. Jara, P. Lopez, D. Fernandez, J. F. Castillo, M. A. Zamora, and A. F. Skarmeta, "Mobile digcovery: discovering and interacting with the world through the internet of things," Personal and ubiquitous computing, 18(2), 323-338, 2014.

[3] A. J. Jara, A. C. Olivieri, Y. Bocchi, M. Jung, W. Kastner, and A.F. Skarmeta, "Semantic web of things: an analysis of the application semantics for the IoT moving towards the IoT convergence," International Journal of Web and Grid Services, 10(2-3), 244-272, 2014.

[4] L. Atzori, A. Iera, G. Morabito and M. Nitti, "The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization," Computer Networks, 2012.

[5] A. M. Ortiz, D. Hussein, S. Park, S. N. Han and N. Crespi, "The Cluster Between Internet of Things and Social Networks: Review and Research Challenges," IEEE Internet of Things Journal, 2014.

[6] Nokia, "Technology vision 2020 Reducing network latency to milliseconds," Nokia Networks white paper, July 2014.

[7] https://www.facebook.com/toyotafriend

[8] https://secure-nikeplus.nike.com/plus/

[9] http://www.lifely.cc/

[10] J. Formo, "A Social Web of Things," April 2012 (available at http://www.ericsson.com/uxblog/2012/04/ a-social-web-of-things/)

[11] V. Beltran, A. M. Ortiz, D. Hussein, and N. Crespi, "A Semantic Service Creation Platform for Social IoT," in Proc. of IEEE World Forum on the IoT, 2014.

[12] "H2020 INPUT Project," [Online]. Available: http://www.input-project.eu/. [Accessed July 2015].

[13] Jain Raj, and Sudipta Paul, "Network virtualization and software defined networking for cloud computing: a survey", IEEE Communications Magazine, 51.11 (2013): 24-31.

[14] M. Nitti, R. Girau and L. Atzori, "Trustworthiness management in the social internet of things," IEEE Transactions on Knowledge and Data Engineering, 2014.

[15] iCore FP7 project, "D2.3 Architecture Reference Model".