A Cloud of Things Framework for Smart Home Services based on Information Centric Networking

Marica Amadeo
and Antonella Molinaro
and Stefano Yuri Paratore
University Mediterranea of Reggio Calabria
DIIES Department, Reggio Calabria, Italy
Email: {marica.amadeo,antonella.molinaro,yuri.paratore}@unirc.it

Albino Altomare
and Andrea Giordano
and Carlo Mastroianni
ICAR-CNR
Rende (CS), Italy
Email: {altomare,giordano,mastroianni}@icar.cnr.it

***Abstract*—Today, the novel Cloud of Things (CoT) paradigm, where Cloud and Internet of Things (IoT) technologies are merged together, is foreseen as a promising enabler of many real-life application scenarios, like the smart home. However, several issues are still debated in the design of CoT systems, including how to effectively manage the heterogeneity of IoT devices and how to support robust and low-latency communications between the cloud and the physical world. In this paper, we present a novel CoT platform that solves such challenges in the smart home domain by leveraging two groundbreaking concepts: *Information Centric Networking* (ICN) and *Fog Computing*. The proposal, called ICN-iSapiens, is a three-layered architecture where an intermediate (Fog) layer, consisting of smart home servers (HSs), is introduced between the physical world and the remote cloud, to support real-time services and hide the heterogeneity of IoT devices. Communication at the physical layer consists of name-based ICN primitives, which facilitate the network configuration and enable simple and effective interactions between HSs and IoT devices. As proof of concept, an experimental testbed is presented and some application examples are described to showcase the advanced capabilities of ICN-iSapiens.**

***Index Terms*—Information Centric Networking, Named Data Networking, Internet of Things, Cloud of Things, Fog Computing**

I. Introduction

In the last few years, the *Internet of Things* (IoT) paradigm rapidly gained ground, by picturing novel ubiquitous computing scenarios where small every-day objects turn into smart things that measure, understand, and modify the environment.

In parallel, with the explosive growth of data traffic and the surge demand of high-quality heterogeneous services, cloud computing has been extended towards the concept of *Fog Computing*, which offers virtualized resouces at the network edge and provides low-latency, high-bandwidth content delivery [1].

Although cloud and IoT originally evolved as separate research fields, they show many complementary aspects that encourage their integration [2]. Indeed, IoT devices - usually resource constrained - can benefit from the virtually unlimited cloud/fog resources; while the cloud can use IoT to enlarge its scope by dealing with advanced services in real life environments [3]. Integrating cloud and IoT and making the resulting *Cloud-of-Things* (CoT) paradigm a reality is however not straightforward. Several issues are still debated, mainly related to the lack of standard protocols, architectures and interfaces that can support the interconnection between heterogeneous smart objects and the cloud.

In this paper, we focus on a representative CoT scenario, the *smart home*, where devices like sensors and actuators are used to remotely monitor and control the house and its appliances for different purposes like energy, lighting, and air conditioning management. Solutions for the smart home are mainly based on proprietary protocols that prevent interoperability, flexibility and extensibility requirements. Standardization efforts are under way [4], but there is still not a leading solution able to address the heterogeneity of devices and guarantee secure, reliable and responsive communications. Notwithstanding their differences, existing solutions usually adopt the network layer the TCP/IP protocol suite (or modified versions for constrained devices like 6LoWPAN [5]), to guarantee global connectivity in the Internet.

In addition, a revolutionary networking model called Information Centric Networking (ICN) has recently attracted the attention of the research community working on data dissemination in different future Internet domains, including the smart home [6]. ICN does not use IP addresses for sending packets between sources and destinations, but forwards messages directly on names that carry application semantics. This clearly simplifies data retrieval and network configuration, since ICN does not need mechanisms (e.g., DNS) to resolve application-level names into network-layer addresses: *a name can be directly used at the network layer for content/service retrieval* [7] [8]. In a few words, ICN well matches the information-centric pattern of many smart home applications (e.g., temperature or energy monitoring), which care about what data to retrieve (or service to request) instead of which node to connect to. Examples of clean-slate ICN smart home systems with real testbeds can be already found in literature [9], [10], [11]. However, large-scale deployments are currently infeasible, since the global connectivity is IP-based.

It is our convincement that the CoT paradigm can help to fill this gap, by guaranteeing full reachability to information produced in ICN smart homes and, in addition, by offering advanced and efficient computing services. Our target in this paper is to integrate ICN and CoT concepts in a common

978-1-5090-4429-0/17/$31.00

framework for smart homes which capitalizes the benefits of both paradigms. Our proposal, called *ICN-iSapiens*, consists of the following innovative features:

1. According to the Fog Computing paradigm, ICN-iSapiens is organized as a three-layered architecture: an intermediate layer, consisting of smart home server(s) (HSs), is introduced between the physical world and the remote cloud. Each HS can execute locally complex tasks and control the home appliances, thus moving a large part of the (real-time) computation as close as possible to the end devices (EDs).

2. The physical layer of ICN-iSapiens consists of ICN-enabled EDs composing the smart home network. EDs act as home data producers or little service providers, they are configured with meaningful, human-readable ICN names that "reproduce" their high-level functionalities. The HS retrieves data and requests services through ICN primitives.

3. To support heterogeneity, flexibility and extendibility, the HS hides the details of ICN EDs through a *virtual name-based representation* and hosts a *multi-agent* software application to monitor and control the EDs.

As proof of concept of the envisioned architecture, an experimental testbed is presented where ICN EDs interact with the HS by leveraging an extended version of the CCN-Lite software [12]; a Java-based agent platform is deployed in the HS to manage home services and communicate with the remote cloud.

The rest of the paper is organized as follows. Section II surveys existing literature on CoT and ICN and motivates our work; Section III describes the ICN-iSapiens framework, while Section IV presents the implemented testbed. Section V shows some application examples. Finally, Section VI concludes the paper.

II. Background and motivation

A. The Cloud of Things vision

The complexity and variety of IoT applications and the large number of heterogeneous elements involved makes IoT data analysis and operation planning a very difficult task.

Cloud computing can solve many problems related to IoT data storage, computation, optimization and presentation. A currently used CoT approach involves two layers: a *physical* layer and a *remote* (cloud) layer. The physical layer sends IoT data to the remote server(s), which process, analyse and store them by guaranteeing availability to multiple consumers. Remote servers can also compute a suitable operation plan and send the result to each device at the physical layer. This approach, however, can experiment some drawbacks when there are constraints on *responsivity* time, that is, when a system needs to react fast to critical events that may overwhelm its integrity and functionality. Communication lag and remote processing can cause delays that a system simply cannot bear [13]. To overcome this issue, recent works embrace the Fog Computing paradigm [1], where cloud services are provided between EDs and traditional remote data centers, at the network edge. Moving the computation as close as possible to the physical resources implies *(i)* reduced network congestion and latency, since big data transfers are avoided, *(ii)* elimination of bottlenecks resulting from centralized computing, *(iii)* improved security as data stays at the network edge and the exposure to malicious attacks is limited.

The framework deployed in this paper adopts the Fog Computing paradigm and introduces, between the physical world and the remote cloud, a distributed intelligence layer that executes almost all real-time control tasks [14], whereas the remote cloud level remains in charge of non-real-time tasks such as offline data analysis or presentation. IoT applications can be deployed with decentralized control functions and the assistance of the cloud to optimize their behaviour. Decentralization is obtained using a distributed multi-agent system in which IoT application execution is carried out through Agents' cooperation at the network edge [15], [16]. The distributed multi-agent system exploits swarm intelligence concepts [17] [18] [19], i.e., it is made up of a population of simple Agents interacting locally with one another and with their environment. Agents follow very simple rules and, although there is no central control structure dictating how individual Agents should behave, interactions among such Agents lead to the emergence of "intelligent" global behaviour, unknown to individual Agents.

B. Information Centric Networking

Today, there is a variety of ICN architectures implementing name-based communications [20]; among them, Named Data Networking (NDN) [21] is probably the most popular approach with a large research community working in different application fields, including the smart home. Compared to IP-based systems, NDN networking looks easier and well suited to IoT applications [7]:

Naming contents, services and functionalities. Content names are hierarchical (sometimes user-friendly) and flexible, i.e., they can include a variable number of components with unbounded lengths. Although traditionally NDN names are related to content objects, they can be used to identify also devices' functionalities and services. For instance, in [6], names describe sensing services, e.g., an Interest with name */sensing/temperature/bedroom* can be used to require the temperature of the bedroom in a smart home.

Easy and robust information exchange. Communication is receiver-driven; it consists of the exchange of two named packets types, Interest and Data, used to request and transfer the content, respectively. Once the namespace is defined and each ED is configured with the names to use, Interests can be sent to retrieve contents or request services, without the need of other information, e.g., devices' network and port addresses.

Strong security. Security mechanisms are embedded by the producer in the Data packet, thus supporting authentication and integrity directly at the network layer and allowing pervasive in-network content replication. Various cryptographic mechanisms can be supported, including lightweight approaches for IoT messages.

Support of multi-party delivery. NDN does not require complex configurations to support unicast or multicast com-

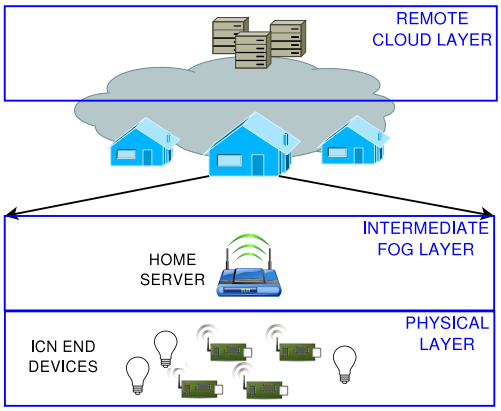

Fig. 1: ICN-iSapiens: actors and cloud/fog/physical layers.

munications: Interests are usually broadcasted over the wireless medium, thus all the nodes working on the advertised name prefix are natively involved in the communication. This facilitates *one-to-one* and *one-to-many* (the so-called *multi-source*) interactions at the same time [22].

C. Information-centric smart homes

So far, different aspects of NDN-based smart home systems have been investigated. Security issues are analysed in [10] where a preliminary lighting control system is designed. NDN names address all the system components and access control to fixtures is performed via authorization policies. Similarly, in [11], a secure building management system is presented, the focus is on data sensor acquisition with encryption-based access control. A home energy management platform to exchange environment sensing and power data is discussed in [9]; in [23] a lighting system is integrated with occupancy detectors and daylight sensors for fully automated operations. All the aforementioned works consider local deployments with well-specific services and no interactions with the cloud or other remote entities over the Internet.

In [22], remote interactions with the smart home are obtained by integrating NDN with the European Telecommunications Standards Institute (ETSI) M2M architecture. The proposal guarantees worldwide connectivity, but it does not integrate with cloud applications, which could realize effective smart home services.

In this paper, we deploy a general-purpose smart home system where NDN interacts with cloud/fog computing to benefit of high-quality, flexible services and global reachability. We focus on architectural aspects to provide the general picture of the system and describe some representative applications implemented in our testbed.

III. The ICN-iSapiens framework

As shown in Fig. 1, ICN-iSapiens is organized into three hierarchical layers which will be detailed in the following.

A. The Physical ICN Layer

At the bottom of the framework, there is the ICN Physical Layer. It includes all the EDs, which deploy sensing and automation tasks in the smart home. They are usually single-function resource constrained devices, like temperature or motion sensors, or light actuators. EDs interact with the HS[1] through the exchange of NDN Interest and Data packets. The main features of this layer are briefly summarized in the following.

Naming design. Each ED is configured on a well-specific hierarchical namespace that describes the *resource(s)* it offers, i.e., sensing measurement(s), automation services. The namespace follows the three-main-components user-friendly structure deployed in [6]. Therefore, we identify *(i)* the task type, *sensing* and *action*; *(ii)* the task subtype, e.g., *energy*, *temperature* for sensing tasks, *light*, *heating* for actuation tasks; *(iii)* one or more task attributes, e.g., the location of the EDs, e.g., bedroom, kitchen.

Service models support. *Pull* and *push* based communications are implemented. Pull based delivery follows the standard NDN packets exchange: the HS sends Interests to obtain sensing data or to request actions. In both cases, Data packets are retrieved that include the sensed information or the result of the action execution, respectively.

Vice versa, push based delivery is implemented via *Interest notifications*, i.e., a special Interest packet where the last component of the hierarchical name is used to carry the information [24], [8]. This is a viable solution in presence of IoT Data, which are usually short. In our design, the HS is configured to accept Interest notifications from the EDs, extract the content from the name field and send a Data packet to acknowledge the reception.

Group-based communications. Thanks to the hierarchical namespace, multi-source communications are also enabled, where a single Interest is used to simultaneously query groups of EDs sharing some part of the same namespace, e.g., the Interest */sensing/temperature* requests the temperature information to every temperature sensor of the house.

B. The Intermediate Fog Layer

The Intermediate Fog Layer includes the HSs, which implement the application logics to monitor and control the house according to *(i)* user preferences, *(ii)* inputs from stakeholders, e.g., service providers and regulation entities, *(iii)* dynamic context-related factors e.g., the energy market price. The HS interacts with the EDs, via ICN, and with the remote cloud, through standard Internet connectivity.

In addition to the software for NDN communications, each HS hosts the multi-agent application and the virtual objects abstraction to perform Fog services.

ICN virtual objects. Each ICN ED is represented at the Intermediate Layer as a *virtual object* (VO), that is a high level standardized description of the device's functionalities. VOs expose EDs by hiding their heterogeneity in terms of technological and networking details, and make their resources easily accessible to the Agents which, in turn, perform the application logics.

[1]We assume the presence of one HS per smart home.

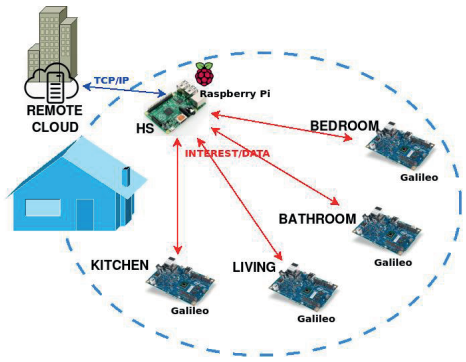

Fig. 2: ICN-iSapiens: testbed.

From a practical perspective, a ICN VO is a *collection of sensing and actuation named functionalities*, and *a collection of standard methods used to monitor/control such functionalities via ICN*. VOs methods call in turn ICN primitives to send named Interest and Data. Hierarchical ICN names well support the VO abstraction: each VO functionality is identified by a meaningful name that is directly used by ICN to interact with the relative ED, without the need of setting-up TCP/IP communications. This clearly simplifies the VO design.

Agents behaviour. Agents use VOs methods to pull monitoring and action tasks, and to be asynchronously notified about some events, i.e., when a resource value changes. Moreover, they may subscribe to complex events defined by boolean rules over groups of functionalities.

To access information through the VO abstraction, VOs and Agents must be co-located in the same HS. Therefore, instead of transferring data to a central processing unit, ICN-iSapiens transfers processes (Agents) towards the EDs. As a consequence, less data needs to be transferred towards remote hosts; local access and computation are fostered in order to achieve good performance (i.e., low latency, real-time services) and scalability (i.e., Fog nodes are geographically distributed, in contrast to centralized Cloud).

C. The Remote Cloud Layer

The Remote Layer includes a remote Cloud platform, which addresses all those activities that cannot be executed by the HSs, e.g., tasks requiring high computational resources or long-term historical data. The data analysis executed by the Cloud can be used for different purposes, including *(i)* to optimize the Agents' operations and their behaviour, *(ii)* to support the demands of external consumer applications, e.g., collected data can be used by energy service companies for reliable forecasting.

IV. Testbed Deployment

As proof of concept of the envisioned framework, we built a demonstrator with low-cost off-the-shelf devices that reproduce an ICN smart home network with CoT capabilities, see Fig. 2.

Devices Features. The HS is implemented over a Raspberry Pi device [25], which is a single-board computer equipped with a SD memory card, an Ethernet interface, and a IEEE 802.11g external interface for wireless communications with the EDs. As operating system we selected Raspbian [26], a free distribution of Debian optimised for the Raspberry Pi hardware. EDs are different kinds of sensors and actuators (e.g., temperature and motion sensors, light actuators) attached to Intel Galileo boards [27]. Galileo is the first 32-bit System-On-A-Chip (SoC) microcontroller board designed to be hardware and software pin-compatible with Arduino shields. Therefore, it is a flexible and cost-effective solution, which can interact with any variety of sensors/actuators and, at the same time, support Linux-based operating systems. We use four Galileo boards in our testbed, each one symbolically located in a different room of the smart home (bedroom, kitchen, bathroom, living) to monitor and control it. Each room can be partitioned in two or more *zones*, each one identified by a number, e.g., both the bedroom and the kitchen have two zones, *zone1* and *zone2*. Each Galileo is one-hop away from the HS and uses IEEE 802.11g shields for wireless communications with it. As operating system, we installed Yocto [28], an open-source complete embedded Linux development environment. Finally, a workstation is used to host the remote cloud applications. It is connected to the campus network and communicates with the HS through standard TCP/IP protocol.

ICN implementation. To enable ICN communications between the Raspberry Pi and the Galileo boards, the CCN-Lite software [12] has been selected. It is a lightweight implementation of the CCNx/NDNx protocols that comes with the fully permissive ISC license and deploys the standard (*static*) content retrieval, based on the *(single) Interest - (single) Data* exchange. The tiny code base has been extended to deal with the real-time production of sensing data and automation services from the EDs. Specifically, each Galileo runs a CCN-Lite instance configured with a set of working name prefixes (e.g., the Galileo in the bedroom works on the prefixes */sensing/temperature/bedroom/*{*zone1,zone2*}, */action/light/*{*on/off/increase/decrease*}*/bedroom/*{*zone1,zone2*}). When the Galileo receives an Interest for a sensing or an automation task, the name lookup is performed. If a matching is found, it launches the correspondent sketch program that drives the execution of the task from the ED(s). Finally, it includes the result into a Data packet that is sent back to the Raspberry Pi. Important modifications on CCN-Lite have been also performed to support the features described in Section III-A (i.e., Interest notification, multi-source delivery).

iSapiens core components implementation. The Raspberry Pi hosts the iSapiens core components operating at the Fog Layer, which consist of: *(i)* the *Agent Server*, a runtime environment for Agents execution and *(ii)* the *VO Container*, an entity that manages the VOs. Each VO is implemented as a collection of VO functionalities, which can be defined through the interface *VOContainer.VirtualObjectFunctionality*. This latter includes a set of methods, which can be defined to control and monitor the ICN EDs. For instance, the *check* method returns sensing information, the *acting* method executes an automation task, the *addStreamListener*

method returns asynchronous sensing notification. Agents use such methods to access the ED resources in a homogeneous fashion and exchange information between each other through asynchronous messaging. In addition, by defining *VOContainer.Rule* objects, boolean rules can be finalized to allow Agents subscribe to specific events (e.g., when the sensed temperature is lower then a threshold, or when the energy consumption reaches a predefined level), and perform complex application logics.

ICN-VO Interface. Thanks to the use of ICN name-based communication, the implementation of the interface between the Physical Layer and the VO abstraction is extremely facilitated. Each physical device can be accessed by directly using the name of its resources, without the need of recall network addresses, port numbers or even layer-2 addresses. The Raspberry Pi hosts a set of C programs (referred to as *ICN send/recv routines*) that take as input ICN names and allow to send Interests and extract information from Data packets or from Interest notifications[2]. *ICN send/recv routines* are invoked, controlled and manipulated from the *VOContainer.VirtualObjectFunctionality* interface.

V. Applications Example

To better understand the behaviour of ICN-iSapiens from a practical perspective, we describe two applications based on a set of monitoring and controlling tasks deployed in the smart home: *(i)* an energy-saving light management application, and *(ii)* a smart door lock application.

As shown in Fig. 2, the smart home consists of four rooms, each one instrumented by the following sensors for the light management application: *(i)* sensors detecting when a person enters or leaves a room, *(ii)* proximity sensors detecting the presence of people in each zone of the room, *(iii)* illuminance sensors. In addition, adjustable brightness lights are included in each zone of each room. The smart door lock application is based on a sensor, embedded on the main door, to detect opening, closing and entrance events. An actuator instead manages the opening and closing of the door. All the mentioned devices work on a specific ICN namespace that identifies their functionalities (see Table I, II for some name examples).

ICN VO design. Each VO abstracts and wraps a certain number of sensors and/or actuators. In this example we define a *Virtual Light object*, whose structure can be replicated for each room of the house to create the Virtual Kitchen Light, the Virtual Bedroom Light, etc. Table I shows the functionalities exposed by the Virtual Kitchen Light, together with the correspondent ICN names. Each functionality of the virtual light object is parametric: the *zone* parameter specifies which area of the room is referred. Similarly, a *Virtual Home Door object* is defined, as shown in Table II.

Agents design. The target of the light management application is to adjust the lights in each room on the basis of people presence/movements and the current illuminance. This logic is implemented in the *LightAgent*. When the application is active, the HS periodically sends Interest packets carrying the relative names, e.g., an Interest with name *sensing/illuminance/kitchen/zone1* is issued to query the illuminance sensor in the zone1 of the kitchen. At the higher layer, the *Virtual Kitchen Light* collects the sensed values and makes them available to the LightAgent. A set of simple rules are defined to support the application logic. For instance, a switch-on operation is issued when the illuminance value is lower than a target threshold set by the user, and the human presence is detected. In this case, the VO *acting* method is invoked and then the actual switch-on command is sent in an Interest packet, e.g., with name *action/light/on/kitchen/zone1*. Moreover, the light brightness is adjusted by considering the number of people in the room and their position.

The target of the smart door lock application (executed by the DoorAgent) is to manage the locking of the main door by considering the number of people in the house at a given time and also the owner preferences, e.g., he/she can order to lock/unlock the door from his mobile phone, or command the locking of the door at midnight.

In summary, there is a LightAgent per room and a unique DoorAgent per home.

Agent deployment, interaction and acquaintance relationships So far, by referring to our testbed scenario, we considered the special case of a single HS in the smart home, therefore ICN EDs always communicate with the node where all the VOs are deployed. However, in a more general case, several HSs can be instantiated, e.g., consider a large villa, a residence or an office building. In such a case, the application design has to take into account that each VO and the EDs enclosed by it must be located in the same HS. For instance, in the case of a large villa, we can identify groups of rooms and assign each group to a different HS. Figure 3 shows the assignment to three HSs in the large villa topology. Agents without connection with any physical part can be located everywhere, even in a remote cloud node.

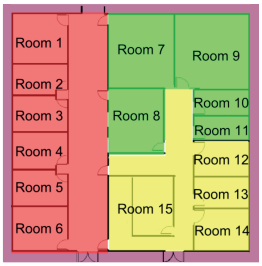

Fig. 3: Rooms assignment to HSs in a large villa topology. Each different color identifies a different HS.

A system component called *Deployer* is in charge to load the Agents upon the Agent Servers, to establish acquaintance relationships among them and to start the application. Acquaintance relationships are necessary to let Agents interact with each other. For instance, consider a HeatingAgent, which manages the heating system by considering the current sensed

[2]We extended the *ccn-lite-peek* utility, already available in the CCN-Lite package, to allow the HS send Interests.

TABLE I: Kitchen Virtual Light.

Functionality	Type	Description	ICN Namespace
near people	Sensing	number of people in the zone	/sensing/people/kitchen/{zone1,zone2}
increase_light	Action	increase light brightness in the zone	/action/light/increase/kitchen/{zone1,zone2}
decrease_light	Action	decrease light brightness in the zone	/action/light/decrease/kitchen/{zone1,zone2}
light_off	Action	set off light in the zone	/action/light/off/kitchen/{zone1,zone2}
light_on	Action	set on light in the zone	/action/light/on/kitchen/{zone1,zone2}
illuminance	Sensing	illuminance in the zone	/sensing/illuminance/kitchen/{zone1,zone2}

TABLE II: Virtual Home Door.

Functionality	Type	Description	ICN Namespace
locking status	Sensing	Boolean (true if the door is closed, false if the door is open)	sensing/door/locking{true, false}
entrance status	Sensing	Integer (Counting the number of people inside the house. The counter increases when a person enters the house, decreases when a person leaves)	sensing/door/entrance{true, false}
open	Acting	Open the door	acting/door/open
close	Acting	Close the door	acting/door/close

temperature and the number of people in the house. When all the people leave and the door is locked, the HeatingAgent should turn off the system.

After loading each Agent in the proper location, the *Deployer* sends acquaintance messages to the HeatingAgent in order to let it know the DoorAgent. Afterwards, the HeatingAgent sends an acquaintance message to the DoorAgent in order to be known by it. Once the deployment phase is completed, the application execution can start. When each person leaves the house, the DoorAgent locks the door and sends a message to the HeatingAgent, which will turn off the heating system.

VI. Conclusion

In this paper, a novel CoT platform for Fog-enhanced smart home services, called ICN-iSapiens, has been defined. The proposal leverages the innovative Fog Computing and ICN paradigms to deploy smart monitoring and control applications in an efficient and effective fashion.

VII. Acknowledgement

This work was partially funded under grant PON03PE_00050_2 DOMUS "Cooperative Energy Brokerage Services", MIUR.

References

[1] F. Bonomi et al., "Fog computing and its role in the internet of things," in *MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[2] A. Botta et al., "Integration of Cloud Computing and Internet of Things: a Survey," *Future Generation Computer Systems*, vol. 56, 2016.

[3] R. Petrolo, V. Loscrì, and N. Mitton, "Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms," *Transactions on Emerging Telecommunications Technologies*, 2015.

[4] A. Kamilaris, A. Pitsillides, and V. Trifa, "The smart home meets the web of things," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 7, no. 3, pp. 145–154, 2011.

[5] Z. Shelby and C. Bormann, "6LoWPAN: the Wireless Embedded Internet," 2009.

[6] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Information Centric Networking in IoT scenarios: The case of a smart home," in *IEEE ICC*, 2015, pp. 648–653.

[7] W. Shang et al., "Named Data Networking of Things," in *IEEE IoTDI*. 2016, pp. 117–128.

[8] M. Amadeo et al., "Information-Centric Networking for the Internet of Things: Challenges and Opportunities," *IEEE Network*, vol. 30, no. 2, pp. 92–100, 2016.

[9] J. Zhang, Q. Li, and E. M. Schooler, "iHEMS: an information-centric approach to secure home energy management," in *IEEE SmartGridComm*, 2012.

[10] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control and NDN," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 2013, pp. 394–398.

[11] W. Shang et al., "Securing Building Management Systems Using Named Data Networking," *IEEE Network*, vol. 28, no. 3, pp. 50–56, 2014.

[12] "Ccn-lite," http://www.ccn-lite.net/, (Accessed 2016.10.7).

[13] A. Giordano, G. Spezzano, A. Vinci, G. Garofalo, and P. Piro, "A cyber-physical system for distributed real-time control of urban drainage networks in smart cities," in *International Conference on Internet and Distributed Computing Systems*. Springer, 2014, pp. 87–98.

[14] A. Giordano, G. Spezzano, and A. Vinci, "Rainbow: an intelligent platform for large-scale networked cyber-physical systems." *UBICITEC*, vol. 2014, pp. 70–85, 2014.

[15] J. Lin, S. Sedigh, and A. Miller, "Modeling cyber-physical systems with semantic agents," in *IEEE COMPSACW)*. 2010, pp. 13–18.

[16] N. Bicocchi, M. Mamei, and F. Zambonelli, "Self-organizing virtual macro sensors," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 7, no. 1, p. 2, 2012.

[17] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999, no. 1.

[18] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. Morgan Kaufmann, 2001.

[19] F. Cicirelli, A. Forestiero, A. Giordano, and C. Mastroianni, "Transparent and efficient parallelization of swarm algorithms," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 2, June 2016.

[20] B. Ahlgren et al., "A Survey of Information-Centric Networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, 2012.

[21] L. Zhang, *et al.*, "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[22] M. Amadeo, O. Briante, C. Campolo, A. Molinaro, and G. Ruggeri, "Information-centric networking for M2M communications: Design and deployment," *Computer Communications*, vol. 89-90, 2016.

[23] U. De Silva et al., "Named Data Networking Based Smart Home Lighting," in *ACM SIGCOMM 2016 Conference*, 2016, pp. 573–574.

[24] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Secure sensing over named data networking," in *Network Computing and Applications (NCA)*, IEEE, 2014, pp. 175–180.

[25] "Raspberry pi," http://www.raspberrypi.org/, (Accessed 2016.10.7).

[26] "Raspbian free operating system," http://www.raspbian.org/, (Accessed 2016.10.7).

[27] "Intel Galileo board," http://ark.intel.com/products/78919/Intel-Galileo-Board, (Accessed 2016.10.7).

[28] "Yocto project," https://www.yoctoproject.org, (Accessed 2016.10.7).