# On the Integration of Information Centric Networking and Fog Computing for Smart Home Services


**Marica Amadeo, Andrea Giordano, Carlo Mastroianni and Antonella Molinaro**

**Abstract** Research on monitoring and control services for smart home and building management is expanding, stimulated by the growing interest in Cloud computing and Internet of Things. In addition to proprietary platforms, a common trend is to connect the smart home network to the Internet and leverage Cloud resources to run the application logic and store historical information. Recently, in many designs, intelligence is introduced at the edge of the home network to support low complexity operations. Interoperability between the different network domains is offered by the TCP/IP protocol suite and its extension for low-power nodes, i.e., 6LoWPAN. In parallel, the revolutionary Information Centric Networking (ICN) paradigm has been recently proposed to support future Internet communications and also data delivery in smart urban ecosystems, including smart home/building services. By leveraging name-based communication, in-network caching and per-packet security, ICN can largely simplify data delivery and service provisioning in instrumented environments. Moreover, by integrating ICN with Cloud technologies, a comprehensive home management system can be built. In this chapter, solutions are presented that rely on ICN for monitoring and controlling the smart environment. The integration of ICN with Cloud/Fog resources is also discussed and a reference architecture is presented as proof-of-concept, together with a preliminary testbed.

M. Amadeo · A. Molinaro
DIIES Department, University Mediterranea of Reggio Calabria,
Via Graziella Feo di Vito I, 89100 Reggio Calabria (RC), Italy
e-mail: marica.amadeo@unirc.it

A. Molinaro
e-mail: antonella.molinaro@unirc.it

A. Giordano (✉) · C. Mastroianni
CNR - National Research Council of Italy, Institute for High Performance
Computing and Networking (ICAR), Via P. Bucci 41C, 87036 Rende (CS), Italy
e-mail: giordano@icar.cnr.it

C. Mastroianni
e-mail: mastroianni@icar.cnr.it



F. Cicirelli et al. (eds.), *The Internet of Things for Smart Urban Ecosystems*,
Internet of Things, https://doi.org/10.1007/978-3-319-96550-5_4

## 1 Introduction

By promoting the global interconnection of billions of smart objects that produce and consume information, the Internet of Things (IoT) opens new opportunities for users, manufactures and companies to deploy smart urban ecosystems [1]. IoT objects range from quite complex systems, like smartphones and tablets, to very resource-constrained devices with slow CPUs and small memory footprints, like sensors and RFIDs. Of course, the management and processing of the huge amount of information generated by such heterogeneous devices, while also providing a common interface to different players and always-on service availability with security and privacy guarantees, is a challenging task. The dominant strategy, which largely simplifies the overall design, leverages Cloud computing technologies to host the application logic. To complement the Cloud technology, the Fog computing paradigm has been promoted [2], which foresees a set of servers at the network edge and offers low-latency services with context awareness. The IoT can largely benefit from real-time interactions with the Fog, while letting big data analytics and long term storage to the Cloud [3].

In this chapter, we show how Cloud and Fog technologies can play a key role in a representative IoT scenario, the smart home, where a variety of monitoring, automation and control functions can be deployed [4]. Smart home applications and, more generally, building management systems (BMSs) have been mainly based on proprietary protocols and specialized hardware, thus missing interoperability, flexibility and extendibility requirements. Today, however, stimulated by the research in IoT and Cloud technologies, there is a growing interest into novel solutions that allow global reachability to services and contents in such environments; IP-based solutions are under-way to facilitate the access to smart home resources and also reduce the installation and management costs, by introducing open standard hardware and software. On the other hand, very recently, smart home solutions based on the revolutionary paradigm called Information Centric Networking (ICN) are appearing. ICN has been proposed as a novel mechanism to improve the communication in the future Internet [5], including smart urban ecosystems based on IoT and Machine-to-Machine (M2M) environments with resource-constrained nodes [6, 7].

By design, ICN can naturally overcome some of the open issues of IP-based solutions in presence of challenging networks [8]. ICN implements name-based routing and leverages application-meaningful names, which can be used to address at the network layer different types of resources, including contents, services and things. In-network caching and consumer mobility are intrinsically supported, and data security is performed at the network layer. As a result, ICN can simplify the design of IoT applications and, coupled with Cloud and Fog computing, can build a comprehensive framework for effectively managing and control the IoT environment.

Therefore, the main target of this chapter is to analyse ICN-based smart homes and their integration with Cloud/Fog technologies. After shortly introducing the ICN paradigm and its exploitation in a IoT context in Sects. 2 and 3 we discuss the applicability of ICN in smart homes and building management systems, by identifying

its native benefits and surveying the related literature. Then, Sect. 4 discusses the most relevant state-of-the-art solutions for the exploitation of IoT and Fog computing technologies in smart home environments. In Sect. 5, we describe the methodologies adopted to integrate the technologies discussed so far, i.e., ICN and Fog computing, in a smart home scenario. Finally, Sect. 6 shows a practical implementation of the ICN-Fog integration by presenting ICN-iSapiens, a framework deployed under an Italian academic/industrial project, and Sect. 7 concludes the chapter.

## 2 ICN-IoT in a Nutshell

ICN was originally conceived as a new paradigm for the future Internet where information is the first class network element and communication is based on unique, persistent, and location-independent content names, used by applications for search and retrieval. Over the years, several ICN architectures appeared that share the same common principles [5], and the model evolved towards a network where names not only identify contents, but also heterogeneous resources, like services and things [9].

In the following, without loss of generality, we refer to a very popular ICN architecture called Named Data Networking[1] (NDN). The NDN project was funded in the US by the National Science Foundation (NSF) in September 2010 and it is in constant development, with contributions from the worldwide research community. NDN uses a hierarchically structured naming scheme, with names in the form of Universal Resource Identifiers (URIs) having a potentially unrestricted number of components. For instance, the temperature values provided by a sensor located in the telecommunication laboratory of the University of Reggio Calabria could be named as */unirc/engineering/tlclab/temperature/*, while the humidity values could be named as */unirc/engineering/tlclab/humidity*, and so on. Communication is based on two named packet types, the *Interest*, used to request a resource (e.g., a content, an actuation command, a generic computation), and the *Data*, which answers the Interest by carrying the outcome of the request. In this way, ICN naturally matches the pattern of many IoT applications, which are interested in the data itself, e.g., the temperature in a specific zone, not in a well-specific data source. Moreover, thanks to the consumer-driven connectionless communication, NDN supports consumer mobility: when the consumer changes its point of attachment, it can simply re-express the unsatisfied Interests, without the need of resuming any previous data session. This is especially useful in many mobile IoT scenarios, e.g., in presence of electric vehicles.

NDN implements per-packet security: the Data includes a digital signature, created by the producer, which binds the name of Data with its content, so that the authenticity can be verified at any time. As a result, each named Data packet is a self-consistent unit, which can be cached by any in-network node, according to its policies and resources, to serve future requests without involving the original sources.

[1] https://named-data.net/.

This is a great departure from the IP model, where, due to the host-centric connectivity, caching can be implemented only in specific nodes, e.g., a Proxy that separates the IoT domain from the Internet. In-network caching is, instead, a key feature of ICN that not only limits retrieval delay and network load, but it is especially useful in presence of resource-constrained nodes, which sleep most of the time and cannot tolerate massive data access.

Caching operations in NDN also respect the freshness requirement that usually characterizes many IoT Data, e.g., a temperature value can vary over time. The original producer can set a *FreshnessPeriod* in each Data packet so that the information is considered stale after that time.

Thanks to the name-based forwarding, NDN also supports native multicast delivery. Specifically, an NDN node is provided with the following Data structures, which are used in the forwarding process: *(i)* a *Content Store* (CS), to cache incoming Data packets; *(ii)* a *Pending Interest Table* (PIT), to store incoming Interests that are not consumed by the Data packets; *(iii)* a *Forwarding Information Base* (FIB), to route Interests towards the content sources. At the Interest reception, a node *N* first looks in the CS: if a name matching is found, then the Data can be sent back immediately. Otherwise, *N* looks in the PIT: if a name matching is found, it means that the same request has been sent, and the node is waiting for the result. Therefore, *N* just updates the existing PIT entry with the Interest incoming interface. Otherwise, *N* looks in the FIB. If a matching is found, *N* creates a novel PIT entry and sends the Interest over the outgoing interface. Otherwise, the Interest is discarded and optionally a negative acknowledgement is sent back.

The Data packet follows the PIT entries back to the requester(s): *N* sends the packet to the interfaces from which the Interest was received, (optionally) caches the data in the CS, and removes the PIT entry. Data without a PIT matching are discarded. Figure 1 summarizes all the forwarding process.

Forwarding rules are defined by the *Routing Information Base* (RIB), which records routing information, used to update the FIB when needed, and by the *Strategy Choice Table* (SCT), which collects a set of named forwarding strategies, associated with the namespaces, manually chosen by the system administrators. This is another important feature of NDN: packets can be treated differently by merely considering their name prefix. Therefore, for instance, all the packets from delay-sensitive emergency applications, e.g., with main prefix/*emergency*, can be forwarded with high priority and over multiple interfaces simultaneously.

## 3 ICN for Smart Homes and Building Management

In the near future, homes and buildings will host a large set of small, resource-constrained devices, like sensors, actuators and controllers, that will participate in different automation and monitoring applications, like smart lighting, smart energy consumption, surveillance, and healthcare. A Gateway can act as the *Home Server* (HS) that controls the devices and bridges the home network to the Internet.

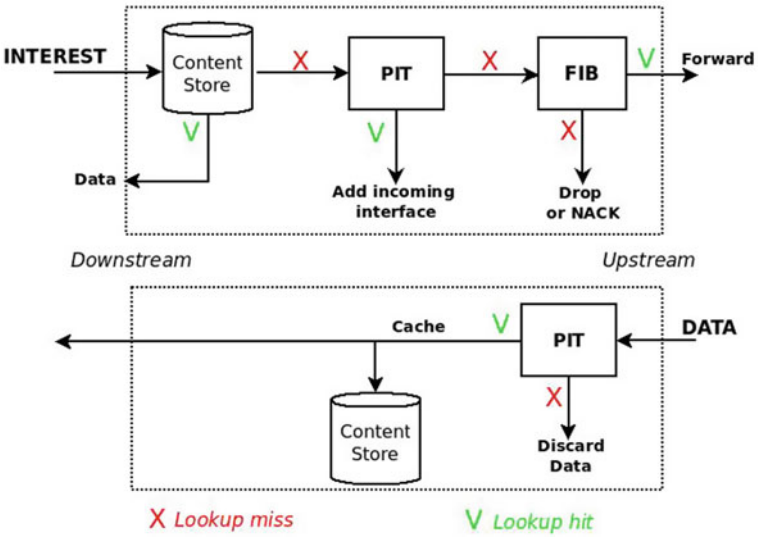


**Fig. 1** NDN forwarding process

ICN holds great promise in instrumented environments by offering the following distinctive features.

**Simplified configuration and management**. By using application-level names, ICN simplifies network configuration and management, since there is no need of addressing each host and updating the address if the node moves, e.g., healthcare applications can be based on wearable devices that move with the patient. Names can be defined that are meaningful for the applications, but also able to support routing scalability [10]. Automatic self-configuration mechanisms, enabling practical ICN deployments on IoT networks, have been also defined [11], where each device is able to derive names locally, by satisfying the requirements of meaningfulness and uniqueness.

**Improved security**. Many home applications involve high sensitive information and need strong privacy, integrity and authentication support. Per-packet security implemented by ICN ensures that the data is secured, independently from the node that serves it and the channel where it travels. Data packets can also be encrypted to ensure access control. Security mechanisms can be extended to Interest packets; this is especially useful in presence of automation applications, like the lighting control, where requests need to be authenticated [12]. Security can also be implemented in presence of low-powered nodes, as in [13, 14].

**Group-based communications**. Home applications frequently originate group-based communications, where multiple data sources are queried at the same time and/or the data is of interest of multiple consumers. Therefore, in addition to the *single-consumer-single-source* communication model, we can identify *single-consumer-multi-sources* (SC-MS), *multi-consumers-single-source* (MC-SS) and

even *multi-consumers-multi-sources* (MC-MS) delivery patterns. Such multicast transmission modes can be supported by ICN without introducing additional signalling, by simply leveraging the forwarding fabric and proper naming schemes [15, 16].

**Easy and efficient data access**. Name-based forwarding coupled with in-network caching can effectively simplify and improve data reachability, even when the data sources are occasionally disconnected, e.g., due to low battery level. Indeed, data in ICN can be delivered to interested consumers without any direct connectivity with resource-constrained IoT devices. To this purpose, proper distributed caching policies can be defined that maintain IoT contents available at any time while devices are in deep-sleep mode [11, 17], by also limiting the energy consumption in the IoT network [18]. The consistency of IoT contents is guaranteed by introducing the *FreshnessPeriod* in the Data packets at the time of creation, so they can be removed by the CS when stale. It is also worth noticing that energy savings are possible thanks to (even small) in-network caching in the IoT network, as experimentally demonstrated in [19]. Indeed, on-path caching decreases the number of devices involved in the data delivery and, at the same time, content producers can sleep more time, since their contents are available in the network.

In the following, by reviewing the related literature, we present the main aspects of ICN-IoT solutions for smart homes and BMSs. The discussion is organized by considering the principal implementation aspects, namely naming, security, caching, and communications patterns.

## *3.1 Data Naming Solutions*

Although there are not a-priori restrictions in the way ICN names can be composed, it is a shared belief that some common conventions must be defined, in order to facilitate application interoperability and re-usage. The most popular, widely accepted, solution is to define user-friendly, hierarchical namespaces, which reflect the application scope and/or the data type offered by IoT devices.

Typically, ICN-IoT names include: *(i)* a root prefix, used to forward the Interest towards the IoT domain (e.g. the smart home), *(ii)* a set of middle components that identify the application type and/or the thing or the data produced by the device, and *(iii)* optionally, a final component that identifies the specific instance of the Data [9].

In [12], a namespace for the BMS at UCLA campus it defined, which reflects the physical location of the devices and the hierarchy of the building structure. A sensor data packet can be named as */ndn/ucla.edu/bms/building/melnitz/studio/1/data/panel/J/voltage/<timestamp>*, where the main prefix *ndn/ucla.edu* indicates the IoT domain at UCLA, *bms* indicates the application type, the components *building/melnitz/studio/1/data/panel/J/voltage/* identify the location of the sensor and its measurements (voltage). Finally, the timestamp identifies a specific voltage instance.

Similarly, in [15], sensing and control operations in the house are expressed through a hierarchical human readable namespace, having as root prefix */homeID/task*.

The component */homeID* is a unique name related to the geographic location of the house and/or to an owner identification number, and it is advertised by the HS in the ICN network thus remote consumers can reach the smart home and its services. The route prefix is followed by the components */type/subtype/location/*, which specify, respectively, the task type (sensing or action), the specific sensing or action task to be performed, and the physical position of the device in the house, e.g., the name */bobHouse/task/action/light/on/kitchen* is used to require the kitchen light fixture to turn on.

It is worth noticing, however, that human-readable names may not be required or permitted in presence of low power, low-bandwidth communications, whereby no humans are involved, and transmission technologies require small MTU (Maximum Transmission Unit). In such case, short naming schemes, e.g., compact ASCII representations, are necessary to leave more space for data payload [19]. An auto-configuration mechanism, with no a-priori human intervention, has been deployed in [11], which derives short name prefixes from the device type and the (unique) node identifier like the vendor ID, and uses timestamps, as version numbers, e.g., */hum/DEADBEEF/1466250645*.

## *3.2 Security*

Producer's digital signature is mandatory in each Data packet to make it verifiable from any node. A *MetaInfo* field in the Data packet header indicates the name of the public key that can verify the signature. Interest packets, instead, are not signed in the standard ICN framework.

When dealing with smart home automation services, however, Interests can be used to request different types of actions, e.g., to switch on/off lights, to open/close windows, and, of course, the packets need to be authenticated to prevent malicious intrusions in the home. The use of *signed Interest*, originally proposed in [12] for lighting control, is today a common practice: the packet is signed by the consumer with a private key, the receiver will perform the action only if it is successfully verified with the consumer's public key.

To manage and assign keys, trusted third parties are needed in the IoT domain. For instance, in [13], a framework for ICN secure sensing is proposed that introduces two entities in the IoT domain: a Configuration Manager (CM) and an Authorization Manager (AM). The CM assigns to sensors the namespace for publishing data, a unique public/private key, and the AM's public key that identifies the root of trust shared by all sensors and applications within the same domain. For each sensor, the AM distributes a signed access control list containing the identities and namespace of applications which are allowed to query it.

Although public key cryptography is widely used in standard ICN implementations, it could not be supported in presence of resource-constrained devices, like small battery-operated sensors and actuators commonly used in the smart home. This is why the research community has defined alternative less-expensive security

mechanisms. NDN-ACE is an actuation framework based on symmetric cryptography that offloads key distribution and management to a powerful third party called Authorization Server (AS) [14]. Consumers and actuators first establish trust relationships with the AS. Then, each actuator delegates access control tasks to the trusted AS by periodically sharing with it a per-service secret key (seed), which is transmitted in an encrypted packet. When the consumer needs to access a service, the AS authenticates its identity and generates an access key for the client. The key is derived from the corresponding service seed and cryptographically bound to the service name and the client identity. The client sends an authenticated Interest that carries two security-related information: *(i)* the HMAC (Hash-based Message Authentication Code) signature signed with the access key, which indicates that the client has been authorized by the AS, and *(ii)* signature information, which allows the actuator—which does not maintain per-consumers key—to recompute the access key from the service seed and finally authenticate the consumer. If the Interest is successfully verified, the actuator executes the command and sends an acknowledgement Data, HMAC-signed with the same access key.

### *3.3 Caching*

To maximize the benefits of in-network caching, proper strategies must be defined in order to limit the energy consumption in the smart home domain by also guaranteeing high content reachability, even if the data sources are in sleep mode.

A Cooperative Caching mechanism, called CoCa, for low-power devices has been deployed in [11]. CoCa leverages link-local broadcast transmissions to maximize data availability in presence of nodes with intermittent connectivity. Specifically, each Data packet generated by IoT nodes is broadcasted in the IoT domain, so that each receiving node can decide to cache it, according to its content capacity and the caching strategy, like random caching or Max Diversity Most Recent (MDMR). By doing so, when an Interest is transmitted in the IoT domain, there is (almost always) an active node with a valid copy of the content. Of course, CoCa can be successfully implemented in small network topologies like a smart home environment, but it suffers from scalability issues when dealing with high density IoT deployments, since it can lead to broadcast storm phenomena.

In this case, high-selective caching mechanisms can be implemented. pCASTING, in [18], is a probabilistic strategy that adjusts the caching probability according to the data freshness and two device features, the battery energy level and the cache occupancy of the node. At the Interest reception, the IoT node computes a caching utility function that simultaneously takes into account the above normalized parameters and computes the caching probability. As a result, pCASTING efficiently stores Data in the network by limiting the energy consumption in the nodes and, at the same time, guaranteeing low content retrieval delays.

There are cases, however, where IoT information does not need to be cached, e.g., because it is privacy-sensitive, or simply not of interest for other consumers.

Consumers and/or producers can request to not perform caching operations by including this information in the Interest or the Data. For instance, in [20], the consumer can set a special bit in the Interest, thus the corresponding content is not cached by intermediate routers.

### *3.4 Communications Patterns*

IoT communication patterns are very heterogeneous: they differ in the number of involved actors, i.e., single/multiple consumers and producers, and in the service model, i.e., push/pull.

Multi-consumer communications (MC:SS) are inherently supported by the ICN forwarding fabric, thanks to Interest aggregation in the PIT. Vice versa, multi-source communications (SC:MS, MC:MS) need slightly modifications in the PIT behaviour to allow that multiple data can be accepted with a single Interest. The work in [16] implements multi-source retrieval by leveraging a proper-defined naming scheme, where sources of the same type publish data under the same principal prefix, e.g., the temperature sensors in the same house share the principal prefix */BobHouse/temperature* and publish Data with names */BobHouse/temperature/bedroom*, */BobHouse/temperature/bathroom*, etc. When the consumer wants to collect more Data with a single request, it sends an Interest with the principal prefix, thus multiple nodes has a partial name matching and answer the request. The pending Interest in the PIT is not deleted after the first Data reception, but it remains active to collect other packets.

So far, we considered a pull-based service model, guided by the consumer, that matches a large set of smart home applications, namely polling-based monitoring and actuation-based applications. However, many sensor-based applications require publish-subscribe communications, where devices send data to subscribers in a periodic or event-based manner.

The so-called *long-lived-Interests*, originally proposed in [21], are designed to support push-based delivery over NDN. The idea is to use Interests as subscriptions, which are not deleted after the first Data reception but remain active in the PIT to allow the forwarding of other packets. The main con of such implementation is the need of maintain soft-state information in each NDN node; however, this is not an issue in small network topologies like the smart home.

## 4 IoT Fog Computing for Smart Homes

In the context of Smart Homes, in the last few years many advancements and challenges have been done concerning several fields: *Home Automation and Domotics* [22], *Energy Optimization* [23], *Activity Recognition* [24], *Ambient Assisted Living* [25, 26], *Indoor Positioning Systems* [27], and *Home Security* [28]. Nowadays

Smart Home applications can be developed exploiting technologies and solutions offered by the emerging research field of the Internet of Things [29], in which many devices have been enhanced with respect to their ordinary role to be proactive and collaborative with other devices [30]. In this background, exploiting IoT for the new era of Smart Home applications [31] represents an important research field both in academia and in industry context. One of the major challenges in the development of these applications is supporting interoperability among various heterogeneous devices and their proper deployment. Thus, the need for new architectures and frameworks—supporting both smart control and actuation—has been identified by many researchers. The research community has presented several proposals regarding middlewares and frameworks for the rapid prototyping of smart environments in general, or smart homes in particular. As an example, authors in [32] presented *Voyager*, a framework conceived for the support to the development of smart environments. Voyager relies on the exploitation of small bluetooth devices, which are user programmable, used to give intelligence to developed systems. The work in [33] introduced *JCAF* (Java Context-Awareness Framework), an extensible framework that supports the creation of context-aware in-home applications. The main components of the framework are *clients*, *actuators*, *monitors*, and *services*. JCAF allows all of them to be dynamically added, updated or removed at runtime. Authors of [34] presented *Gaia*, a middleware that supports the control of heterogeneous resources in a physical space. Gaia allows the developers to see collections of individual devices as a whole by introducing programmable *active spaces*. Active spaces are programmable through a scripting language called *LuaOrb* [35]. The *Syndesi* framework was introduced in [36]. Syndesi exploits Wireless Sensor Newtorks to create personalized smart environments. The environment augmented with the Syndesi framework allows to identify people and perform specific actions based on people profiles.

Considering that urban environments usually concern large city areas, a distributed approach appears as a "natural" solution [37]. Conversely, most of the actual smart city implementations rely on centralized approaches, where a big amount of data is collected in a common data center that processes such data and provides services to citizens, or schedules actuations on physical infrastructures [38]. Moreover, many services are developed as independent monolithic blocks, and their interaction is not properly considered. In a more distributed approach, benefits can arise from the exploitation of the so-called *Fog computing* paradigm [39–41]. With Fog computing, rather than processing information in a monolithic Cloud layer, the computation is pushed close to the data sources or, said in another way, at the "edge" of the network. Other benefits of this computing paradigm include: *(i)* a faster reactivity to events, *(ii)* a better exploitation of the communication bandwidth, as data is processed locally and only aggregated information is propagated across the system, *(iii)* an increase in reliability and scalability, since the Fog computing paradigm fosters the use of distributed algorithms.

## 5 Integrating ICN with Fog and Edge Computing

Fog computing and ICN are key technologies for future IoT applications. On the one side, the Fog is able to extract knowledge from IoT raw data and deploys the application logic that process IoT data and monitors and controls IoT environments. It is complemented by the Cloud, when needed. On the other side, ICN can largely improve data retrieval and service access, mainly thanks to name-based forwarding and in-network caching.

It is worth observing that, when dealing with IoT environments, some research works refer to the *Edge computing*, instead of the Fog computing. Indeed, edge computing should focus more toward the things side [42], with IoT end-devices (EDs) acting both as data consumers and producers, and the intelligence and processing power installed even into simple, low-cost devices like programmable automation controllers or Raspberry-Pi boards. At the same time, some research works use the terms Fog computing and Edge computing interchangeably, while others consider the Fog a broad concept that includes also Edge computing [43].

In the following, we use the term Fog for the sake of brevity, although the discussion can consider aspects of Edge computing.

Although research on the integration between Fog/Cloud computing and ICN is still at the beginning, two main approaches can be identified.

The first one considers a full ICN-based network model and assumes that name-based computing requests are resolved by in-network nodes, possibly close to the client. In this case, ICN nodes dynamically act like Fog servers: computing capabilities are not statically hosted by well-defined edge servers, but are distributed in ICN routers. When the task cannot be resolved in the network, the request is forwarded to the Cloud.

The second approach, instead, considers ICN deployments only at the network edge, and uses Fog servers as computing nodes that also bridge the ICN domain with the IP-based Internet by leveraging proper *translation* mechanisms.

In the following, we discuss the mentioned approaches by surveying the related literature.

### *5.1 Full ICN-Based Design*

The authors in [44] present Named Function Networking (NFN), an ICN extension where consumers request computations over contents (e.g., a video compression) and the requests are routed towards in-network nodes that will execute them. Functional programs are in the form of λ-calculus expressions and the ICN forwarding machine is augmented with a λ-expression resolution engine, which processes all the Interests that have the postfix name component */NFN*. Therefore, unlike the traditional Fog computing, where Fog servers are fixed hosts with dedicated functionalities, in NFN the network becomes a distributed and dynamic Fog platform, where each content

router, depending on its local capabilities, can act as computing unit. However, how to distribute the processing load between NFN nodes is still an open issue.

In [45], NFN is customized for IoT networks. The proposal, called PIoT (Programmable IoT), is an application layer solution, statically installed in the more powerful nodes of the network, which consists of the following services: *(i)* Expression Resolver, which resolves the λ-expression into sub-tasks; *(ii)* the Expression Pusher, which creates new Interests to fetch the IoT data needed to execute the task; *(iii)* the Expression Evaluator, which executes the task and generates the resulting Data packet(s). In [46], the same authors extend PIoT with a computation service management (CS-Man) that enables IoT devices to offer in-network computing operations, based on their own capability. In CS-man, a network entity acts as Service Manager (SM), which maintains a Service Repository (SR) with the names of the Service Executor (SE) nodes. So, when a client needs a processing task, it can retrieve the identity of the SE(s) from the SR.

Similarly to NFN, a framework for in-network function execution called Named Function as a Service (NFaaS) has been deployed in [47]. NFaaS supports processing with lightweight virtual machines in the form of named *unikernels*. A *Kernel Store* in ICN nodes stores function codes, and decisions on which functions to execute or delete are based on a score function that takes as inputs the requests popularity and the latency/bandwidth requirements of the applications. Nodes advertise the functions they can execute according to a routing protocol, so that task requests are forwarded towards the more appropriate node. The authors consider as an example two forwarding strategies, for delay-sensitive and bandwidth-hungry services, respectively, and demonstrate that functions move to the right direction (i.e., delay-sensitive services towards the edge of the network, and bandwidth-hungry services towards the core).

In [48], ICN supports service provisioning in challenged networks with intermittent connectivity. Services are built as containers coupled with a semantic naming scheme, thus they can be easily requested and cached in the network. The ICN framework is enhanced with a Delay Tolerant Networking (DTN) interface, which communicates with an underlying DTN implementation that handles intermittence by encapsulating Interest and Data packets in DTN bundles.

## 5.2 Hybrid IP-ICN Design

Large-scale clean-slate ICN implementations are still far from being a reality. The Internet is currently based on the TCP/IP protocol suite, and ICN protocols will be more likely implemented in a virtual environment or as an overlay over the TCP/IP transport. Vice versa, edge domains, like the smart home, can easily deploy new implementations from scratch. Therefore, in the following we focus on a hybrid scenario where the IoT domain is ICN-based, while the Internet is IP-based. The IoT domain leverages a Fog layer to support local storage and processing capabilities. In addition, Fog servers bridge the ICN domain to the IP-based Internet and

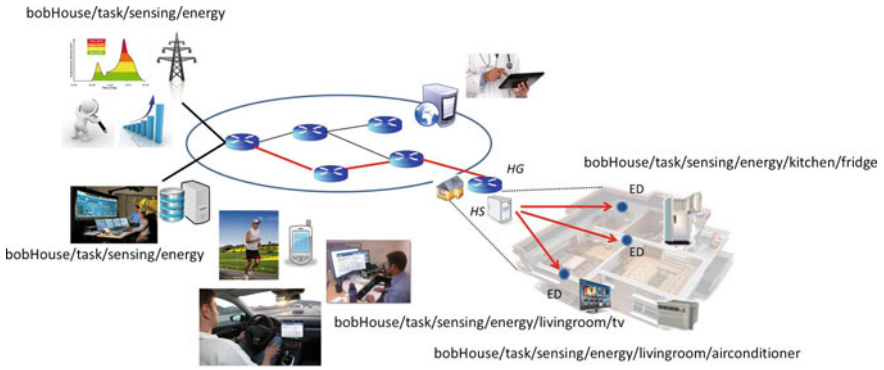


**Fig. 2** Reference scenario

guarantee full reachability to IoT information, in a way that is independent from the technological and networking details of IoT devices.

A Fog computing-based gateway (FOGG) that bridges the Internet with ICN-IoT domains is proposed in [49]. The FOGG gateway is a powerful node that collects information from sensors by using the lightweight ICN implementation called *ccn-lite* (www.ccn-lite.net/), and makes it available to external users by running some translation mechanisms. It also controls actuations, with inputs from remote consumers, and manipulates raw data to extract meaningful information.

In the context of smart homes, Fog (or Edge) computing capabilities can be implemented in the smart home server (HS), which natively monitors and controls the environment, by interacting with ICN end-devices, and with remote clients and the Cloud through the Internet. Therefore the HS acts also as a Home Gateway (HG), as shown in Fig. 2.

The result is a three-layered architecture with a physical ICN layer, the Fog as intermediate layer and the Cloud as upper layer. In the next section, we present a real implementation of this framework, which has been described in a previous paper [50]. The core component of the framework is *iSapiens*, an IoT platform that allows the development of general cyber-physical with edge computing capabilities [51, 52].

## 6 The Case of the ICN-iSapiens Framework

### *6.1 Architecture Design*

Figure 3 shows the three-layered ICN-iSapiens framework.

The *Physical Layer* includes all the EDs, which deploy sensing and automation tasks in the smart home and communicate with the Home Server via ICN. Each ED

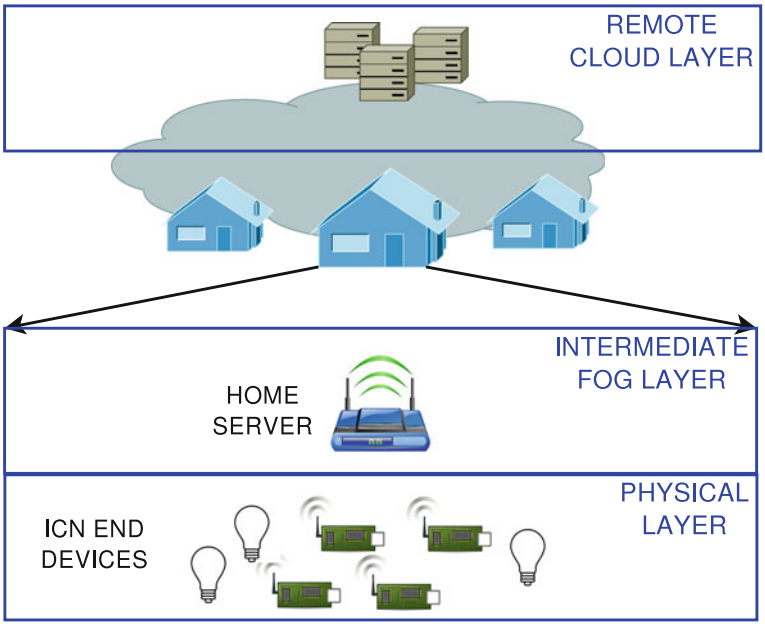


**Fig. 3** ICN-iSapiens framework

is configured on a specific hierarchical namespace that describes the resource(s) it offers, i.e., sensing measurement(s) and automation services. The namespace identifies *(i)* the task type, sensing or action; *(ii)* the task subtype, e.g., energy and temperature for sensing tasks, light and heating for actuation tasks; *(iii)* one or more task attributes, e.g., the location of the EDs like bedroom or kitchen. Moreover, the name hierarchy is structured in a way that multi-source communications are 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/temp" requests the temperature information to every temperature sensor of the house.

The *Intermediate Fog Layer* includes the Home Server (HS) that hosts the ICN protocols to communicate with EDs and the TCP/IP protocol stack to communicate with remote entities through the Internet. Moreover, the HS hides the details of ICN EDs through a name-based *virtual objects abstraction* and hosts a multi-agent software application to monitor and control the EDs.

Each ICN ED is indeed represented as a virtual object (VO), that is a high-level standardized description of the device's functionalities that hides its heterogeneity in terms of technological and networking details. Each ICN VO can include a collection of sensing and actuation *named functionalities*, and a collection of standard *methods* used by Agents to monitor/control such functionalities via ICN. Agents use VOs methods to pull monitoring and action tasks, and to be asynchronously notified about some events; indeed, VOs methods call ICN primitives to send named Interest and

Data. There is a one-to-one correspondence between VO functionalities and ICN names, so the VO name is directly used by ICN to interact with the related ED.

Finally, the Cloud layer addresses all those activities that cannot be executed by the HSs, e.g., tasks requiring high computational resources or long-term historical data. Moreover, the data analysis executed by the Cloud can be used to optimize the Agents' behaviour.

## 6.2 Testbed

As shown in Fig. 4, an ICN-iSapiens demonstrator with low-cost off-the-shelf devices has been deployed. The HS is implemented over a Raspberry Pi device (www.raspberrypi.org/), equipped with an Ethernet interface, for Internet communications, and a IEEE 802.11g external interface for wireless communications with the EDs. *Raspbian* (http://www.raspbian.org) is the selected operating system. EDs are different kinds of sensors and actuators, like temperature and motion sensors, light actuators, attached to Intel Galileo boards, with *Yocto* (http://ark.intel.com/products/78919/) as operating system.

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

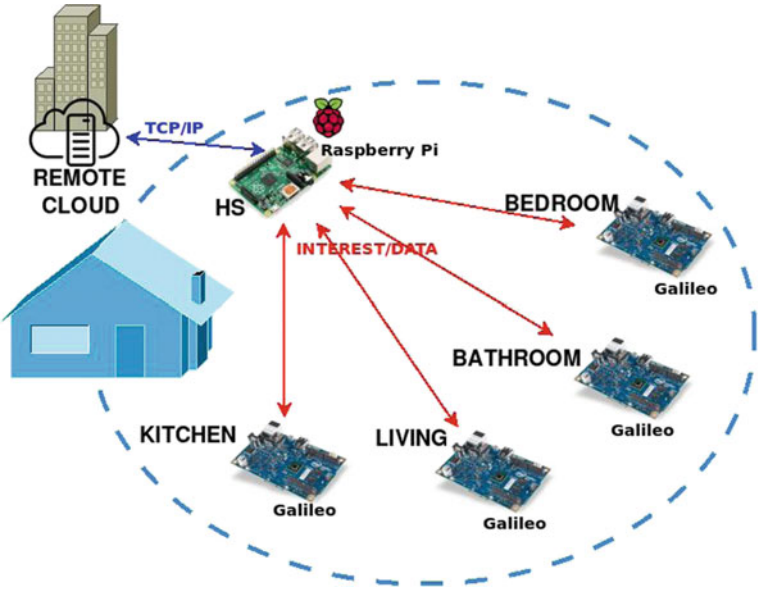


**Fig. 4** ICN-iSapiens testbed

by a number, e.g., zone1 and zone2. Each Galileo is one-hop away from the HS and uses IEEE 802.11g shields for wireless communications with it. 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 communications are deployed with the CCN-Lite software (www.ccn-lite.net/), a lightweight implementation of the CCN/NDN protocols that has been properly extended to support smart home services. The Raspberry Pi also 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. A set of methods are defined to allow Agents to control and monitor ICN EDs, e.g., the *check* method returns sensing information. Complex application logics, based on specific rules, can also be defined. VOs are named resources and such names can be directly used to access them at the physical layer via ICN. The Raspberry Pi hosts a set of C programs that take as input ICN names and allow to send Interests and extract information from Data packets.

As application example, we consider a lighting system, which adjusts the lights in each room on the basis of people presence/movements and the current illuminance. At this purpose, we define a *Virtual Light* object, whose structure can be replicated for each room of the house thus creating the Virtual Kitchen Light, the Virtual Bedroom Light, and so on. Each virtual light object includes the following action functionalities: *(i) light-on* and *light-off*, to switch on or off the light; *(ii) increase-light* and *decrease-light*, to increase/decrease the light brightness in the zone. In addition, two sensing functionalities are deployed: *(i) illuminance*, to identify the illuminance in the zone and *(ii) near-people*, to identify the number of people in the zone.

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. The *Virtual Kitchen Light* collects the sensed values and makes them available to the LightAgent, which controls the lights. For instance, it switches on the light 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 a switch-on command is sent in an Interest packet, e.g., with name action/light/on/kitchen/zone1.

## 7 Conclusions

This chapter has discussed the technological solutions that integrate the Information Centric Networking (ICN) and the Cloud/Fog computing paradigms so as to combine their features in an Internet of Things scenario. The Fog computing paradigm has been promoted to complement the Cloud technology so as to provide intelligence at the edge of the network and offer low-latency services with context awareness. In parallel, ICN has been proposed as a novel mechanism to improve the communication

in the future Internet, including IoT and Machine-to-Machine (M2M) environments with resource-constrained nodes.

In this chapter we focused on a representative IoT scenario, i.e., the smart home, where a variety of monitoring, automation and control functions are required. A reference architecture is presented as proof-of-concept, together with a preliminary testbed. The deployed framework, ICN-iSapiens, conceived under an Italian academic/industrial project, shows a practical implementation of the ICN-Fog integration.

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

## References

1. G. Fortino, P. Trunfio, *Internet of Things Based on Smart Objects: Technology, Middleware and Applications* (Springer International Publishing, 2014). https://doi.org/10.1007/978-3-319-00491-4
2. F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in *Proceedings of the First Workshop on Mobile Cloud Computing (MCC)* (2012), pp. 13–16
3. A. Guerrieri, V. Loscri, A. Rovella, G. Fortino, *Management of Cyber Physical Objects in the Future Internet of Things* (Springer International Publishing, 2016). https://doi.org/10.1007/978-3-319-26869-9
4. A. Giordano, G. Spezzano, A. Vinci, Rainbow: an intelligent platform for large-scale networked cyber-physical systems, in *UBICITEC* (2014), pp. 70–85
5. B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A survey of information-centric networking, IEEE Commun. Mag. **50**(7) (2012)
6. M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R.L. Aguiar, A.V. Vasilakos, Information-centric networking for the Internet of Things: challenges and opportunities. IEEE Netw. **30**(2), 92–100 (2016)
7. M. Amadeo, O. Briante, C. Campolo, A. Molinaro, G. Ruggeri, Information centric networking for M2M communications: design and deployment. Comput. Commun. **89**, 105–116 (2016)
8. W. Shang, Y. Yu, R. Droms, L. Zhang, Challenges in IoT networking via tcp/ip architecture, tech. rep., NDN Project, Technical Report NDN-0038 (2016)
9. W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, L. Zhang, Named data networking of things, in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)* (2016), pp. 117–128
10. W. Shang, Q. Ding, A. Marianantoni, J. Burke, L. Zhang, Securing building management systems using named data networking. IEEE Netw. **28**(3), 50–56 (2014)
11. O. Hahm, E. Baccelli, T.C. Schmidt, M. Wählisch, C. Adjih, L. Massoulié, Low-power Internet of Things with NDN and cooperative caching, in *Proceedings of 4th ACM Conference on Information-Centric Networking (ICN)* (2017)
12. J. Burke, P. Gasti, N. Nathan, G. Tsudik, Securing instrumented environments over content-centric networking: the case of lighting control and NDN, in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (IEEE, 2013), pp. 394–398
13. J. Burke, P. Gasti, N. Nathan, G. Tsudik, Secure sensing over named data networking, in *2014 IEEE 13th International Symposium on Network Computing and Applications (NCA)* (IEEE, 2014), pp. 175–180
14. W. Shang, Y. Yu, T. Liang, B. Zhang, L. Zhang, NDN-ACE: access control for constrained environments over named data networking, tech. rep., NDN Project. Technical Report NDN-0036, Revision 1 (2015)

15. M. Amadeo, C. Campolo, A. Iera, A. Molinaro, Information Centric Networking in IoT scenarios: the case of a smart home, in *2015 IEEE International Conference on Communications (ICC)* (IEEE, 2015), pp. 648–653
16. M. Amadeo, C. Campolo, A. Molinaro, Multi-source data retrieval in IoT via named data networking, in *Proceedings of the 1st International Conference on Information-Centric Networking* (ACM, 2014), pp. 67–76
17. M.A.M. Hail, M. Amadeo, A. Molinaro, S. Fischer, On the performance of caching and forwarding in information-centric networking for the IoT, in *International Conference on Wired/Wireless Internet Communication* (Springer, 2015), pp. 313–326
18. M.A. Hail, M. Amadeo, A. Molinaro, S. Fischer, Caching in named data networking for the wireless Internet of Things, in *2015 International Conference on Recent Advances in Internet of Things (RIoT)* (IEEE, 2015), pp. 1–6
19. E. Baccelli, C. Mehlis, O. Hahm, T.C. Schmidt, M. Wählisch, Information centric networking in the IoT: experiments with NDN in the wild, in *Proceedings of the 1st International Conference on Information-Centric Networking* (2014), pp. 77–86
20. G. Acs, M. Conti, P. Gasti, C. Ghali, G. Tsudik, Cache privacy in named-data networking, in *2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)* (2013), pp. 41–51
21. A. Carzaniga, M. Papalini, A.L. Wolf, Content-based publish/subscribe networking and information-centric networking, in *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking* (ACM, 2011), pp. 56–61
22. A.Z. Alkar, U. Buhur, An Internet based wireless home automation system for multifunctional devices. IEEE Trans. Consum. Electron. **51**, 1169–1174 (2005)
23. J. Serra, D. Pubill, A. Antonopoulos, C. Verikoukis, Smart HVAC control in IoT: energy consumption minimization with user comfort constraints. Sci. World J. (2014)
24. P. Rashidi, D.J. Cook, Com: a method for mining and monitoring human activity patterns in home-based health monitoring systems. ACM Trans. Intell. Syst. Technol. **4**, 64:1–64:20 (2013)
25. N. Pavón-Pulido, J.A. López-Riquelme, J. Ferruz-Melero, M.A. Vega-Rodríguez, A.J. Barrios-León, A service robot for monitoring elderly people in the context of ambient assisted living. J. Ambient Intell. Smart Environ. **6**(6), 595–621 (2014)
26. F. Cicirelli, G. Fortino, A. Giordano, A. Guerrieri, G. Spezzano, A. Vinci, On the design of smart homes: a framework for activity recognition in home environment. J. Med. Syst. **40**, 200 (2016)
27. P. Richter, M. Toledano-Ayala, G.M. Soto-Zarazúa, E.A. Rivas-Araiza, A survey of hybridisation methods of GNSS and wireless LAN based positioning system. J. Ambient Intell. Smart Environ. **6**, 723–738 (2014)
28. L. Sang-hyun, J.-G. Lee, M. Kyung-il, Smart home security system using multiple ANFIS. Int. J. Smart Home **7**, 121–132 (2013)
29. D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac, Internet of Things. Ad Hoc Netw. **10**, 1497–1516 (2012)
30. G. Fortino, A. Guerrieri, W. Russo, Agent-oriented smart objects development, in *2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (May, 2012), pp. 907–912
31. I. Bierhoff, A. van Berlo, J. Abascal, B. Allen, A. Civit, K. Fellbaum, E. Kemppainen, N. Bitterman, D. Freitas, K. Kristiansson, *Smart Home Environment* (COST, Brussels, 2007)
32. A. Savidis, C. Stephanidis, Distributed interface bits: dynamic dialogue composition from ambient computing resources. Pers. Ubiquitous Comput. **9**, 142–168 (2005)
33. J.E. Bardram, R.E. Kjær, M. Pedersen, Context-aware user authentication—supporting proximity-based login in pervasive computing, in *UbiComp 2003: Ubiquitous Computing* ed. by A. Dey, A. Schmidt, J. McCarthy, (eds.), vol. 2864 of Lecture Notes in Computer Science (Springer Berlin Heidelberg, 2003), pp. 107–123
34. M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, K. Nahrstedt, A middleware infrastructure for active spaces. IEEE Pervasive Comput. **1**, 74–83 (2002)

35. R. Cerqueira, C. Cassino, R. Ierusalimschy, Dynamic component gluing across different componentware systems, in *Proceedings of the International Symposium on Distributed Objects and Applications, DOA '99* (Washington, DC, USA) (IEEE Computer Society, 1999), pp. 362–371
36. O. Evangelatos, K. Samarasinghe, J. Rolim, Syndesi: a framework for creating personalized smart environments using wireless sensor networks, in *Proceedings of the 2013 IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS '13* (Washington, DC, USA) (IEEE Computer Society, 2013), pp. 325–330
37. C. Mastroianni, E. Cesario, A. Giordano, Efficient and scalable execution of smart city parallel applications, in *Concurrency and Computation: Practice and Experience*, Aug. 2017. Early view, http://dx.doi.org/10.1002/cpe.4258
38. A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities. IEEE Internet of Things J. **1**, 22–32 (2014)
39. P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, E. Riviere, Edge-centric computing: vision and challenges. SIGCOMM Comput. Commun. Rev. **45**, 37–42 (2015)
40. F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA) (ACM, 2012), pp. 13–16
41. M. Yannuzzi, F. van Lingen, A. Jain, O.L. Parellada, M.M. Flores, D. Carrera, J.L. Perez, D. Montero, P. Chacin, A. Corsaro, A. Olive, A new era for cities with fog computing. IEEE Internet Comput. **21**, 54–67 (2017)
42. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges. IEEE Internet of Things J. **3**(5), 637–646 (2016)
43. M. Chiang, S. Ha, I. Chih-Lin, F. Risso, T. Zhang, Clarifying fog computing and networking: 10 questions and answers. IEEE Commun. Mag. **55**(4), 18–20 (2017)
44. M. Sifalakis, B. Kohler, C. Scherb, C. Tschudin, An information centric network for computing the distribution of computations, in *Proceedings of the 1st ACM International Conference on Information-centric Networking* (2014), pp. 137–146
45. Y. Ye, Y. Qiao, B. Lee, N. Murray, Piot: programmable IoT using information centric networking, in *2016 IEEE/IFIP Network Operations and Management Symposium (NOMS)* (2016), pp. 825–829
46. Q. Wang, B. Lee, N. Murray, Y. Qiao, Cs-man: computation service management for IoT in-network processing, in *IEEE Signals and Systems Conference (ISSC)* (2016), pp. 1–6
47. M. Król, I. Psaras, NFaaS: named function as a service, in *Proceedings of the 4th ACM Conference on Information-Centric Networking* (2017), pp. 134–144
48. C.-A. Sarros, A. Lertsinsrubtavee, C. Molina-Jimenez, K. Prasopoulos, S. Diamantopoulos, D. Vardalis, A. Sathiaseelan, Icn-based edge service deployment in challenged networks, in *Proceedings of the 4th ACM Conference on Information-Centric Networking* (ACM, 2017), pp. 210–211
49. S.S. Adhatarao, M. Arumaithurai, X. Fu, FOGG: a fog computing based gateway to integrate sensor networks to internet, in *IEEE Teletraffic Congress (ITC 29)*, vol. 2 ( 2017), pp. 42–47
50. M. Amadeo, A. Molinaro, S.Y. Paratore, A. Altomare, A. Giordano, C. Mastroianni, A cloud of things framework for smart home services based on information centric networking, in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)* (2017), pp. 245–250
51. F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, An edge-based platform for dynamic smart city applications. Future Gener. Comput. Syst. **76**, 106–118 (2017)
52. F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, G. Ruggeri, Edge computing and social internet of things for large-scale smart environments development. IEEE Internet of Things J. (2017)