Journal of Ambient Intelligence and Humanized Computing (2019) 10:2645–2656
https://doi.org/10.1007/s12652-018-0954-5

ORIGINAL RESEARCH

Towards a more reliable and scalable architecture for smart home environments

Valère Plantevin[1] · **Abdenour Bouzouane**[1] · **Bruno Bouchard**[1] · **Sebastien Gaboury**[1]

Received: 15 March 2018 / Accepted: 20 July 2018 / Published online: 9 August 2018

Abstract
The Internet of things (IoT) has profoundly changed the way we imagine information science and architectures and smart homes are an important part of this domain. Created a decade ago, the few existing prototypes use technologies of the day, forcing designers to create centralized and costly architectures raising issues concerning reliability, scalability as well as ease of access, which cannot be tolerated in the context of assistance. To answer this specific problematic, we propose, in this paper, a new kind of smart home architecture based on a highly distributed environment. We showed that our novelty brings a lot fewer Single Point of Failure (SPoF) to ensure the best reliability achievable in this kind of smart environment. Moreover, we tested our solution with a custom-messaging protocol specifically developed for this kind of architecture to demonstrate that it can achieve at least the same performance in terms of messages per second and quantity of exchanged information than our old centralized smart home architecture.

Keywords IoT · Reliability · Smart home

1 Introduction

The Internet of Things (IoT) profoundly remodeled our relationship with the science of information and especially the different sources of information we now allowed in our day to day life. One of the applications of such concept is the smart home, which became the subject of numerous researches (Cook et al. 2012; Ghayvat et al. 2015; King and Jansen 2005) in the field of Ambient Intelligence (Amb. I). This concept referring to a tendency to miniaturize a set of electronic devices (sensors and effectors) in order to integrate them into any object of everyday life (lamp, refrigerator, etc.) in a transparent way for the person. The aim behind this idea is to supply punctual assistance to the occupants according to the gathered information and to the history of the accumulated data.

The vast majority of work in the smart home domain focuses on the activity recognition problem in order to assist the inhabitant with a potential dementia often caused by an advanced age (Patterson et al. 2003; Augusto and Nugent 2006; Roy et al. 2013). Nevertheless, none of them seems to propose a standard architecture that provides both high-reliability and scalability capabilities at a relative low-cost. And still, high-reliability has to be a mandatory feature of such architecture since the assistance is vital for the inhabitant with a potential dementia. Moreover, as the disease can stay for decades, any work on architecture must take the scalability parameter into account since many new sensors can be installed or improvements performed during the illness evolution. Finally, the low-cost aspect has to be taken into account as the vast majority of the aging population whose suffering from diseases or mental illness already struggle with many medical expenses (Bureau 2016; Alzheimer's Association 2017) consequently we have to keep our solution as cheap as possible.

Here, we introduce a new kind of smart home architecture providing both reliability and scalability based on low-cost smart sensors. To achieve this objective, the first issue we ran into is the difference (protocol, standard, etc.) between

✉ Valère Plantevin
valere.plantevin1@uqac.ca

Abdenour Bouzouane
abdenour_bouzouane@uqac.ca

Bruno Bouchard
bruno_bouchard@uqac.ca

Sebastien Gaboury
sebastien_gaboury@uqac.ca

1 UQAC, 555 boulevard de l'Université, Chicoutimi, QC, Canada

all the possible entities in the environment. In fact, we can have an infinite number of different transducers (i.e. sensors and actuators) without any common way to interact with them. The second issue in this kind of work is the communication. A highly reliable and scalable architecture means infrastructure without using Single Point of Failure (SPoFs) like centralized servers, which ease the communication a lot. We already answered this specific problem in a previous work (Plantevin et al. 2017) where we defined a new way to communicate without SPoFs in a Smart Home but we have to conduct more tests in the final architecture we present in this paper. It should be noted that the main focus of this paper is to describe the architecture (hardware and communications) and not how to realize activity recognition and provide services on top of it. It provides you with a scalable and reliable architecture and a way to get the data but not a way to process it.

The rest of this paper is divided in four successive parts. First, we will explore the existing architectures in the field of Smart Homes. Next, we will describe in depth the infrastructure we propose here. Tests on and discussion of them will come after in another part. Finally, a conclusion will end this paper and present some future works we will accomplish.

2 Existing architectures

During the last two decades, many Smart Homes have been built inside laboratories or in research context inside real habitations (Cook et al. 2003; Giroux et al. 2009; Bouchard et al. 2014; Hu et al. 2016). They all implemented a different kind of infrastructure, causing the apparition of different families of architectures, each of them having advantages and drawbacks, that we have to study. This section is divided in three parts, each one describing a kind of architecture. We begin with the Mesh-based family implemented in CASAS (Hu et al. 2016), Smart* (Barker et al. 2012) and other experiments (Zhihua 2016; Zou et al. 2011). Then we will study the OSGI (a.k.a Open Services Gateway initiative) architectures with the works of Lin et al. (2008); Novák and Binas (2011); King and Jansen (2005), and finally, we will describe the industrial family of infrastructure as implemented in LIARA and DOMUS laboratories as well as in LiSA and House_n projects (Colombiano Kedowide 2014; Intille et al. 2006).

2.1 Mesh-based architectures

Many works in the literature have been focusing on the mesh networks and more precisely the IEEE-802.15.4 norm (Callaway et al. 2002; Gutierrez et al. 2001; Montenegro et al. 2007). The smart home is not an exception with the work from CASAS (Hu et al. 2016), Smart* (Barker et al. 2012) and other authors (Zhihua 2016; Zou et al. 2011). Even if all these works have a few differences, they are all similar to the CASAS architecture that we describe in this subsection.

With a main interest in price and ease of installation, CASAS (Cook et al. 2012) the Smart Home in a box, was firstly developed in 2012. As depicted in Fig. 1, it is based on a four elements architecture. The first element in this architecture is the transducers themselves, communicating through a Zigbee mesh where every node relay the information to its neighbors by using the ZigBee protocol. Next, a ZigBee bridge converts these low-level ZigBee events to high-level XMPP messages and relay them to the messaging service. This last one allows other services to easily integrate the infrastructure and use the transducers. By default, there are two services which are the Storage and the Intelligence. The first one stores all the messages occurring in the environment by using the so-called Scribe Bridge. The second one brings artificial intelligence inside the home by monitoring the energy and recognizing the activities carried out inside the house in order to enable a possible assistance.

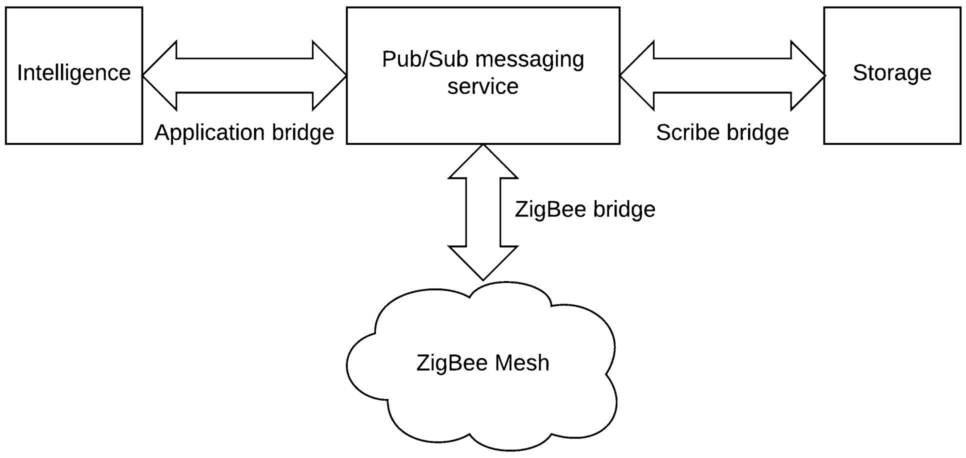

Fig. 1 The CASAS architecture

CASAS, and in a general manner all the mesh-based architectures, have two main benefits, which are the price and the ease of installation. With 2765 US dollars presented in a detailed summary by the authors, their solution seems to be the cheapest available, which is really a great achievement. To prove that their solution is easy to install, they ask peoples aged from 21 to 62 to install it in their houses and it requires only an hour to set up the whole environment, which clearly demonstrates the ease of the installation. Nevertheless, this architecture has one great defect regarding the reliability. Indeed, the whole infrastructure is polluted by Single Points of Failure (SPoFs) like the ZigBee bridge or the Application Bridge that can stop the assistance or the Pub/Sub-messaging service, which is a sensitive component.

2.2 OSGI architectures

Some authors (Lin et al. 2008; Novák and Binas 2011; King and Jansen 2005) tried to answer the smart home scalability problem by using an OSGI based architecture. These works share the same features even if the GatorTech initiative (King and Jansen 2005) was realized twenty years ago. Consequently we only describe here the GatorTech project, however, all the conclusions made on this particular work can be done on the other.

Funded in Florida, Gator Tech (King and Jansen 2005) is a project conducted to prove the feasibility of a low-cost smart home where the integration of new transducers will be easy. To achieve their goal, the authors created an OSGI (OSGi Alliance 2016) based architecture, sums up in Fig. 2, where each sensor or actuator has a simple memory containing the driver to communicate with it. This driver is sent when the transducer is powered up during a registration process to an OSGI service definition. Behind this service definition concept stands an abstraction layer that allows to create basic services providing highly abstracted data for immediate consumption [e.g. "Humid" instead of 95% of humidity for a humidity sensor] or the combination of basic services in a composite service (e.g. create a voice recognition service on all the different microphone services). The work achieved by King and Jansen (2005) allows developers and researchers to create artificial intelligence without any knowledge of the underlying infrastructure and with only highly abstracted data which greatly simplify the development.

Despite the usage of OSGI on a single server that creates a SPoF, which is a problem in a high reliability architecture, this family of architecture has some really good advantages that have been made possible mainly by the embedding of a small piece of logic inside transducers. First of all, the automatic transducer registration really helps the scalability of such an environment (e.g. add new sensors or replace some of them). Secondly, this infrastructure, with its high

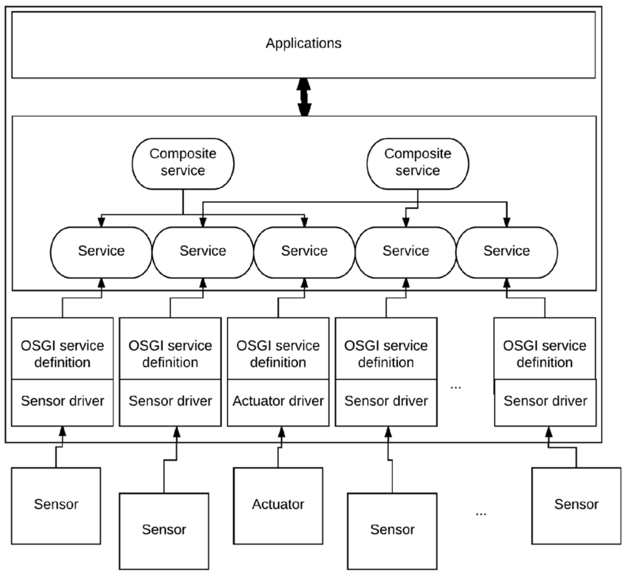

Fig. 2 The Gator Tech architecture

abstraction of the data generated, helps the developers and researchers to conceive and build applications. For example, it is straightforward to enable the air conditioning when the temperature is "Hot". However, it is more complicated when the decision is only based on the microcontroller value since it depends on the hardware (i.e. the microcontroller or the temperature sensor). Finally, the price of such infrastructure is as low as possible as every transducer is designed to be the most affordable possible by using Atmega128 as the main processor unit which is a low-cost platform (Drumea et al. 2005).

2.3 Industrial architectures

In the beginning of the smart environments, some of them were built around material inherited from the industrial automation domain. These infrastructures are present in a various number of works, from the smart car with the LiSA project (Colombiano Kedowide 2014) to the smart homes with, among others, the LIARA, DOMUS and House_n projects (Giroux et al. 2009; Bouchard et al. 2014; Intille et al. 2006). They are all pretty similar in terms of realization, advantages and drawbacks. Consequently, we describe here the architecture present in the LIARA and DOMUS laboratory but every conclusion made of these can be easily made on the others.

In order to study how a smart environment can help people with cognitive deficiencies, LIARA and DOMUS laboratories created a very similar architecture, represented in Fig. 3, to test their algorithms and solutions (Bouchard et al. 2012). Built from industrial grade material, they use a limited number (four) of Advantech islands (Advantech

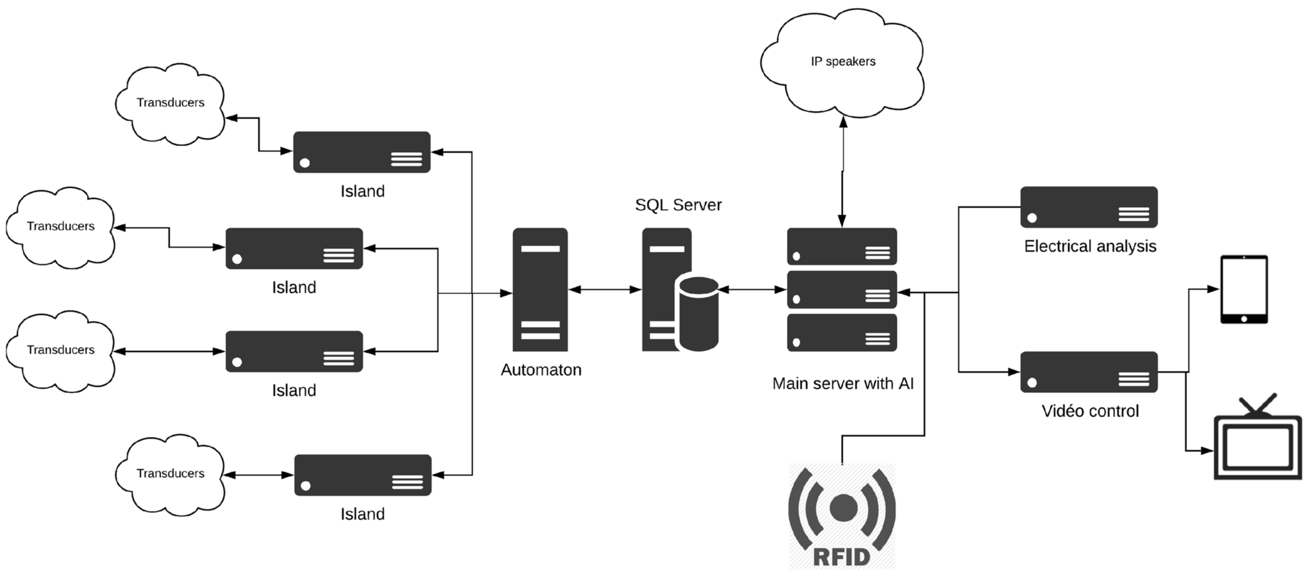

Fig. 3 The LIARA and DOMUS architecture

2016) to agglomerate transducers present in the environment. These islands communicate only over Modbus TCP an old protocol not understandable by every application. Consequently, an Advantech automaton is in charge of getting back, the values of the sensors or changing the values of the effectors. A bidirectional communication, through a relational database hosted on a SQL Server, between the main server and the automaton allows applications (in this case artificial intelligence) to update the state of the effectors or read the values of sensors. Finally, complex transducers like IP speakers, electrical analysis, video control or RFID readers are directly connected to the main server which update the database with the values read from these specific sensors and update these effectors when their values are changed in the database.

The industrial hardware and the hyper-centralization are two main characteristics in these architectures that bring some interesting features. The first one means that all the elements have been highly tested for a continuous use in far much harder environments than just a house (e.g. production line in a factory). Therefore it demonstrates an excellent component reliability and offers strong guarantees to the user even if the smart home has to operate at all times. Moreover, the highly centralized architecture greatly eases the interaction between applications, such as artificial intelligence, and the environment. Indeed, all the values coming from sensors and all the actuator controls end up in the same relational database, which is a widespread means of sharing data across applications.

Even if the industrial material brings some advantages, it suffers from two main drawbacks, which are the proprietary material and the price of this one. The first introduces black boxes in a research environment since the communications between all these pieces of hardware often rely on proprietary libraries that can impact future evolution. Concerning the price of this kind of architecture, based on the hardware presented by Bouchard et al. (2014) and the price of it, we were able to compute the total price of the chain Island-Automaton-Main Server. With 2000 dollars each island (Advantech 2016), 1500 dollars the automaton (Advantech 2016) and 4000 the server (Dell 2016), the architecture reaches 13,500 dollars without any transducers or backbone structure (e.g. networking, cooling for the server, maintenance). Such a high price cannot be tolerated especially in the case of the sick population who already have high medical costs. Finally, the highly centralized architecture presented here creates many Single Points of Failure (SPoFs) like the automaton, the Islands and the main server which hosts both the AI and the SQL server. Thus, if one of these SPoFs fails, at least a quarter of the environment and its assistance will fail too. Moreover, these points represent some serious bottlenecks in the architecture preventing a real scalability.

2.4 Conclusion on existing architectures

To conclude, we can extract the common features of these architectures presented in this section. The first and most obvious one is that the majority of the components composing these infrastructures are transducers. Moreover, it seems that embed some intelligence and communication abilities like in CASA or Gator Tech in them help to ease the installation and the scalability. On the other hand, all these smart home share the same drawback, which is the centralization.

This characteristic creates many single points of failure that can lead to a complete stop of the artificial intelligence or worse, the whole environment. In other computer science domains, this particular issue has been solved ten years ago by using redundancy, clusters and distributed computing (Chu-Sing and Mon-Yen 2000; Lu et al. 2006; Mon-Yen and Chu-Sing 2001; Schroeder et al. 2000).

If we have to describe an ideal architecture we can say, after the cases CASAS and Gator Tech, that it must be composed of many smart transducers easing the scalability of such architecture. Another domain of the computer sciences that seems to be close to this description is the Internet of Things (IoT). In the last one, already used in Smart Homes (Atzori et al. 2010; Liu et al. 2011), a multitude of smart objects communicate in a uniform manner and generates a huge amount of data often associate to "Big Data" (Chen et al. 2014). In this last case, it is not conceivable to handle the information in a centralized way anymore, even if we use server clusters. It is more appropriate to use decentralized methods relocating the intelligence as close as possible to the units composing this huge data pool (Chen et al. 2014; Hey et al. 2009).

3 Proposed architecture

The whole contribution of this paper is a more reliable and more scalable architecture for smart homes. It is summed up in Fig. 4 and will guide our explanation during all this part.

One way to enhance the reliability is to remove the Single Points of Failure (SPoFs) in the architecture. In our old infrastructure, the SPoFs were the islands, the automaton and the main server. In order to remove them we have to provide another way to gather, stream and process the information from all the sensors and to enable the actuators update. In order to achieve such an objective, we decided to upgrade our transducers from simple passive ones to smart ones as represented in part A and D in Fig. 4.

The notion of smart transducers is already clearly defined in the IEEE norm 1451.4 (IEEE 1998). To sum up this last one, we can say that a smart transducer is an entity providing more features than the ones mandatory to generate a good representation of the controlled quantity. Some of these attributes can be sensor identification, a process to simplify the installation or the maintenance, network interfaces or the coordination and synchronization with other entities (Lewis et al. 2004). In order to simplify further tests, developments and adaptations of our solution, we divide the transducer in two different parts. The first one is the sensing or actuating unit, which we will call the transducer unit. It makes the interface between real world (e.g. temperature, lights control, Twitter API, etc.) and high-level data for the further process (e.g. temperature is 25.3 °C, light switched on or there was 26 re-tweet of a post). The second one is the smart unit and is responsible for communications inside the environment, making decisions based on transducer values and messages from other smart transducers and program the transducer unit, depending on the different sensors or actuators connected to it. The communication between these two entities is allowed via a simple serial link in case of hardware transducers or a ZMQ inter-process communication (ZeroMQ 2016) in case of software transducers (e.g. APIs consumption). All these elements are summed up in Fig. 4. We will now discuss in more details each part of this architecture beginning with the smart unit followed by the transducer one. Finally, we will depict how these smart units can communicate together in a high-reliability manner.

3.1 Smart unit

Represented in part **C** of the Fig. 4, the smart unit is the main computing piece of hardware making our transducer a smart one. It is in charge of both the communication with the whole environment and the computation of some sort of intelligence based on transducer values. Moreover, it eases the installation of the different kind of hardware we can find in a Smart Home by flashing and updating embedded drivers if the transducer unit have to communicate with hardware sensors or actuators. This driver is defined by two distinct files encapsulate inside a Zip archive. The first one is the embedded firmware to flash into the microcontroller via the serial link if there is such a piece of hardware (i.e. hardware transducers). The second one is a description file formatted in JSON and containing two main elements : the configuration of the communication with the transducer unit and the JSON send by this last one with each key associated to a small description of the value.

The schema of the configuration file is presented in Fig. 5. In this last one, the first element is the communication configuration defined by the first JSON object. It contains three fields : port, the baud rate and upload. The first two, allow the communication through serial, the baud rate is an integer number and the port is a string corresponding to a serial port on the smart unit operating system. The third item, the upload value, is a string representing the command line to execute in order to upload the firmware to the transducer unit. We decided to use this simple representation to let the user define the kind of upload process he has depending on the hardware on which is implemented the transducer unit. In case of software transducers, the baud rate and upload parameters disappear and the port contains the ZeroMQ Inter-Process Communication address. The second object in the driver description is an array of different objects describing every value send by the transducer unit. This description must contain at least an identifier (*id* here), which is a unique number, its name to enable ease the understanding

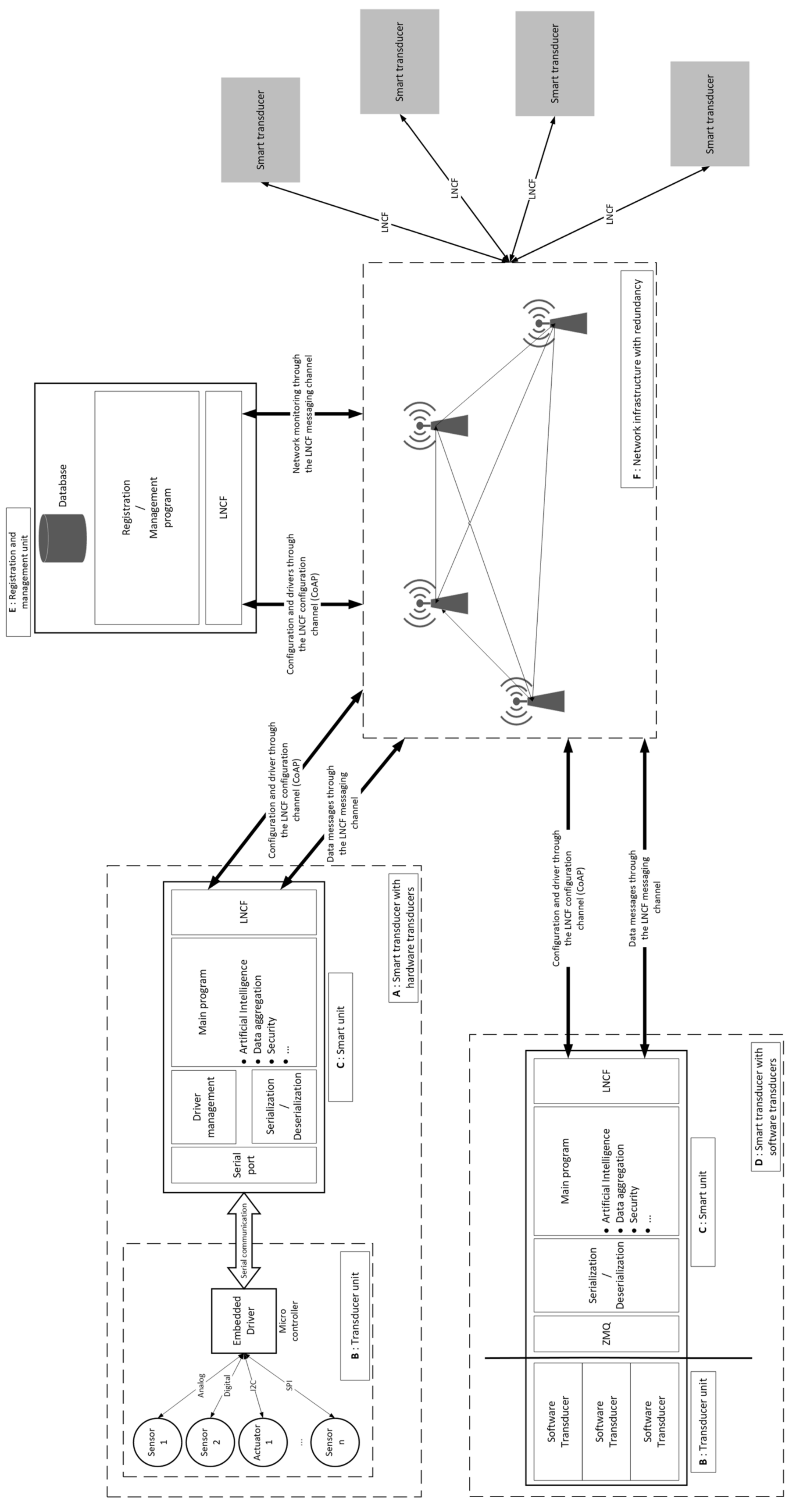
Smart transducer
Smart transducer
Smart transducer
Smart transducer
LNCF
LNCF
LNCF
LNCF
E : Registration and management unit
Database
Registration / Management program
LNCF
Configuration and drivers through the LNCF configuration channel (CoAP)
Network monitoring through the LNCF messaging channel
F : Network infrastructure with redundancy
Configuration and driver through the LNCF configuration channel (CoAP)
Data messages through the LNCF messaging channel
A : Smart transducer with hardware transducers
LNCF
Main program
Artificial Intelligence
Data aggregation
Security
…
Driver management
Serialization / Deserialization
Serial port
C : Smart unit
Serial communication
Embedded Driver
Micro controller
B : Transducer unit
Analog
Digital
I2C
SPI
Sensor 1
Sensor 2
Actuator 1
…
Sensor n
Configuration and driver through the LNCF configuration channel (CoAP)
Data messages through the LNCF messaging channel
D : Smart transducer with software transducers
LNCF
Main program
Artificial Intelligence
Data aggregation
Security
…
Serialization / Deserialization
ZMQ
C : Smart unit
Software Transducer
Software Transducer
Software Transducer
B : Transducer unit

Fig. 4 The whole architecture proposed in this paper

```
{
    serial : {
        port : STRING,
        baudrate : NUMBER,
        upload : STRING,
    },
    data: [
        {
            id : NUMBER,
            name : STRING,
            type : "CONTINUOUS"|"NOMINAL",
            values : [STRING],
            actuator: BOOLEAN
        }
    ]
}
```

Fig. 5 The JSON schema of the driver description file

and communication and a type, which is rather *continuous* or *nominal* depending on the kind of value this transducer deals with. Moreover, there are two optional fields, which are values, an array of string defining the different values possible in case of a nominal sensor, and actuator, a boolean saying if this transducer is an actuator or not (by default this value is false).

When a smart unit first boot (i.e. during its installation, not after a reboot), it generates a unique identifier and register itself to a centralized entity represented in E on the Fig. 4. This entity is a small computer (e.g. a Raspberry or even a phone) that only keep a track of which smart transducer is present in the Smart Home. This entity is not needed for the network to properly work it just allows to easily emit alert if a smart transducer stopped working (i.e. stop communicating since a long time) and the last but not least, it allows to link a smart transducer to its driver, stored inside this entity. Consequently, we need this platform during the installation of new smart transducers or if we want to deeply change the configuration of one of them (e.g. change a temperature sensor for a pressure one, or upgrade a driver). Theoretically, this central entity can be hosted in the cloud to guarantee high-availability and reliability. However, since many smart home networks are in intern loop (i.e. the transducer network has no Internet) for safety reasons, this entity can be hosted locally inside a computer or a phone or a tablet.

3.2 Transducer unit

Represented in part B in the Fig. 4, the transducer unit is nothing other than a piece of hardware or software that generates high-level data from low-level ones for the smart unit.

In case of hardware transducer units, it is a simple microcontroller connected to hardware sensors and actuators and embedding a piece of software defined by the smart unit known as a driver. This one, reads and interprets the values of the different sensors or actuators connected to the unit. Finally, it sends high-level values (e.g. 23.5 °C instead of 256 the value from the sensor) formatted by using a SSV (i.e. Semi-column Separated Value) representation to the smart unit through a Serial Communication. To ease the development of such embedded software, we decided to use the Arduino platform as it is well-known, supported on many microcontrollers in the market, really easy to develop and allows the use of many libraries already developed by the community. Moreover, each transducer unit must implement a way to program its microcontroller through the serial communication like by the usage of a programmer [e.g. FTDI (Future Technology Devices International) chip]. Finally, even if we allow the possibility for a transducer unit to possess more than one hardware transducers, it is a best practice to only bind one hardware sensor to one transducer unit to avoid the apparition of a Single Point of Failure inside the smart transducer.

Concerning the software transducer units, they are services running inside the smart unit and generating high-level data from lower-level APIs. Useful for testing purposes, they also allow to consume APIs from Internet or other services inside the house. This feature permit the generation of high-level data from outside the house of from appliances that do not implement our smart transducer architecture and integrate them inside artificial intelligence decisions. These software-defined transducers communicate with the smart unit through an Inter Processus Communication pipe managed by ZeroMQ (2016) that only use a simple address in a string representation to define the communication.

3.3 Communication

All our smart transducers are connected together via a network infrastructure implementing a form of redundancy represented in F on the Fig. 4. Nevertheless, a reliable communication between them is not trivial. Indeed, many architecture use communication mechanisms that introduce single point of failure inside the infrastructure (e.g. the automaton in LIARA and DOMUS or the OSGI server in Gator Tech or every broker based messaging system). To answer this problematic, we decided to use an IP messaging protocol based on multicast we already defined in a previous work name Light Node Communication Framework (LNCF) (Plantevin et al. 2017).

Like File Transfer Protocol (FTP) (Postel and Reynolds 1985), our protocol relies on two channels depending on the type of data that transit. The first one, called *configuration channel*, use Constrained Application Protocol (CoAP) (Shelby et al. 2014) a well-known protocol in machine to machine communication to ensure reliability during the data transfer. This property let make sure that configuration values, which are low-frequency but critical information, are properly received by the entity that need to be configured. Thus, we propose to use this specific channel to dispatch drivers and configurations coming from the central entity in our network.

In addition to its *configuration channel*, LNCF implements a messaging channel over multicast UDP based on a Publish/Subscribe (Pub/Sub) mechanism. The use of multicast UDP ease the removal of any Single Point of Failure since this protocol enables the communication from one to many without anything else other than an IP infrastructure. Furthermore, this channel makes possible three important features : high frequency data exchange, network discovery and encryption. Even if this channel is less reliable than the first one, it allows to exchange information identified by a subject at a very high frequency (i.e. more than 1000 messages per second). This characteristic is mandatory in any application that need to exchange information in order to construct datasets or deals with Big Data like a smart home do. Consequently, we will use this channel to exchange data messaging in our architecture. Besides, the network discovery mechanism embedded in LNCF makes easy for every smart transducer in the infrastructure the registration process since it can query the network to get the IP address of the central unit through a protocol it already uses for communication. Moreover, the encryption process already implemented in LNCF ensures the security of every message transiting in the network. This point is a must have for an architecture that deals with personal information like the one proposed in this paper. Finally, LNCF using only UDP packets, it can be used in any hardware network protocol implementing UDP like allowing the usage of our architecture on different network infrastructures like mesh-based with ZigBee IP, Bluetooth based with 6LowPan or WiFi and Ethernet.

4 Tests and discussion

To validate our new architecture, we performed some tests on it. We first verified the latency of this infrastructure in order to ensure that it is at least as good as the old one which allowed us to gather every transducer values in a delay between 250 ms and 1 s. During this latency test, we add smart transducers to confirm the scalability of our solution. Next, we conduct some reliability tests to see how our infrastructure endures power problems or the termination of different parts of it. Finally, we have cost the price of our whole architecture to compare it with CASAS and our old infrastructure. But first, we will present the material and the infrastructure we used to conduct those tests.

4.1 Material and infrastructure used

In order to conduct these tests, we first had to implement our smart transducer architecture and then provide a reliable network infrastructure to allow our devices to communicate. We will first describe our implementation of the smart transducer presented in this work and then we will explain the infrastructure which allows us to perform these tests.

For this test, we choose some Open Source hardware to implement our solution to be the more reproducible possible. The smart unit is implemented on a Raspberry Pi Zero W, even if, in a final version of this architecture, it should be embedded directly inside every transducer to ease the installation since it would be a major issue to install many Raspberry inside the home. This platform ensures us enough general purpose inputs outputs (GPIO) to communicate with the hardware transducer units while providing us enough computing power when we will implement some intelligence on our prototypes. Concerning the hardware transducer unit, we decided to use ESP 32 Thing from Sparkfun since it embeds enough inputs/outputs to be interesting and it is fully compatible with the Arduino platform. We use both of these platforms (even if the ESP 32 already implements Wi-FI capabilities, and the Raspberry PI a GPIO) to demonstrate the fact that the serial link between the smart unit and the transducer one was powerful enough to communicate data between the two entities. We connected them through the USB plug on the Raspberry Pi and program them, in the driver, to generate a random number of random types of sensors from 5 to 20 at 2 kHz in order to simplify the test since the real data did not matter for us. The problem we had here is that we did not have enough ESP 32 Thing for every Raspberry Pi Zero we had. Consequently, we tested the driver behavior and inter-unit communication on 5 hardware prototypes and we used 25 software transducers on other Raspberry Pi Zero W. These units scaled from simple message displaying inside a terminal or random number generation to twitter consumers for the most demanding of them. These 30 smart transducers make a pretty good first test for our architecture even if 25 of them have a simulated transducer unit. The whole implementation was made in C++ with the help of the Boost libraries.

The infrastructure is only composed of two wireless routers (LinkSys WRT1900AC). We put them in Wireless Distributed Services modes in order to create two wireless access point to the same network allowing the different smart transducers to choose the wireless router with the strongest signal strength. These two access points are connected together through an Ethernet link. Moreover, a laptop (MSI GT62VR) and a Raspberry Pi 3 are connected to them via Wi-Fi. The Raspberry Pi 3 will play the role of the central unit where the different smart transducers register themselves during the installation and the laptop will play the role of a scientist computer who wants to retrieve datasets from the infrastructure in order to measure latency in it. The final infrastructure is depicted in Fig. 6. Of course, every other application normally communicating over this network has been shut down to minimize the traffic of the older infrastructure since we used its backbone.

4.2 Latency and scalability tests

Normally, the smart transducers do not communicate raw data through the network since they have the computing capabilities to process them and make intelligent decisions. But for the need of this experiment and because some scientists may want to gather datasets to make further experiments, the smart unit only relayed the raw data, associated with its timestamp, on the messaging channel of the LNCF protocol (Plantevin et al. 2017) we use to communicate between the smart transducers. The laptop, connected to the network, gathered all this data, reorder them in a dataset and compute the latency of the whole infrastructure to see what maximum sampling rate we can achieve and what is the delay between the generation of a data and its reception. Moreover, we slowly scaled the whole architecture beginning with the 5 hardware smart transducers and by adding five by five other transducers to attain 30. These results combined with the overall latency give us a pretty good metric on how our solution scales. Moreover, in order to add some metrics to our tests, we decided to report the percentage of utilization of the access points, laptop wireless card in the best and worst case.

The results of this test are summed up in Table 1. In this table, each row represents a number of smart transducers in the whole architecture. The columns represent the sampling rate asked and configured via the configuration channel of the LNCF protocol. Inside each cell, stays the average latency in milliseconds during a 1 minute of dataset recording. This latency is computed by subtracting the creation time to the reception one. We can see that the latency values do not move a lot and this even if we add devices or increase the asked sampling rate. This conclusion is confirmed when we compute the average and standard deviation on all the table, which are 2.95 ms for the first and 0.55 ms for the second. Regarding the percentage of use of the different network devices, we found that it scales from 0.5 to 40.1% for the laptop network card, from 0.2 to 35.6% for the wireless router on which the laptop is connected and from 0.2 to 18.5% for the other one. These numbers are for 1 unit at 1 Hz and 30 units at 1 kHz, which correspond to the minimum and maximum of network utilization.

These results show us some interesting facts about our solution. First, it allows us to retrieve all the transducers state at 1 kHz without any problem except a little network latency of 3 ms. This number is a huge improvement over our last architecture that sends us data every 250 ms in the very best case (i.e. 4 Hz). However, it must be noted that

Table 1 Latency and scalability results of our solution. Each row representing a different number of units inside the architecture and each cell representing the overall latency in milliseconds

Units	1 Hz	10 Hz	100 Hz	500 Hz	1 kHz
5	3.0	2.7	3.1	3.1	2.9
10	2.8	2.6	3.4	2.6	3.2
15	3.1	2.4	2.6	2.8	2.6
20	3.1	3.3	3.6	3.5	2.4
25	3.2	5.1	2.3	2.2	2.8
30	2.9	2.6	2.2	3.4	2.9

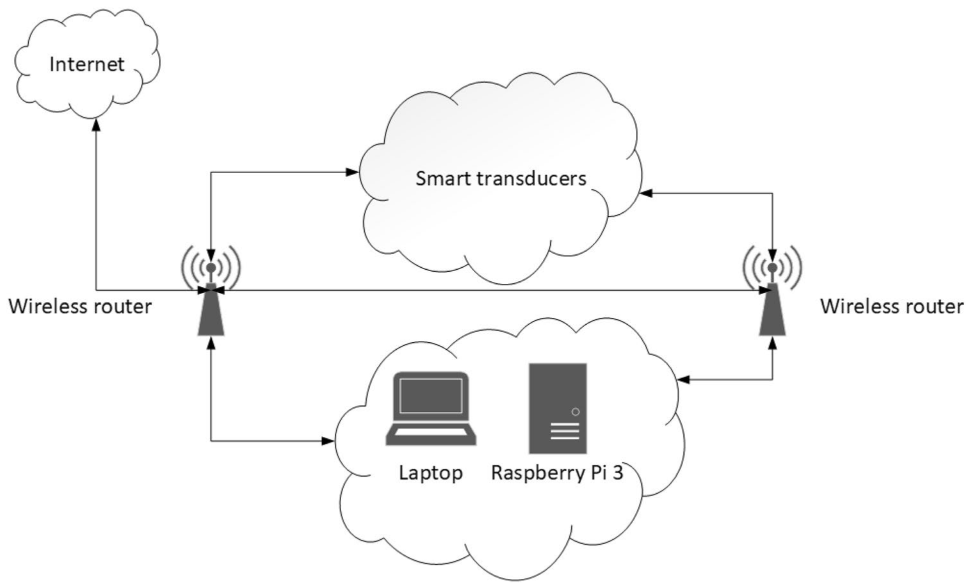

Fig. 6 The full test infrastructure

these numbers can change depending on the technologies used to create the infrastructure since some platforms can take much more time to send a packet or much less than a Raspberry Pi. Secondly, the relatively fixed network latency and the low percentage of use of the different networking appliances show us that even with 30 smart transducers at 1 kHz we are a long way from the saturation of the infrastructure even if this one is a relatively simple one with only two wireless router we can find in any house nowadays. On the other hand, these results also demonstrate that we have to conduct more tests with more smart transducers in order to see how the whole architecture reacts to a 100% of network usage.

4.3 Reliability test

We affirmed that our infrastructure is more reliable and we had to confirm this affirmation. Consequently, we try different scenarios on our solution. The first one is the electrical shut down of one of the two access points (we switch off the breaker on this power line). The second one is the removal of the central unit represented by the raspberry pi 3 that allows to keep a track on every transducer in the network. Finally, we decide to switch off every beaker (one every minute) in the house ending it by a wireless router to switch them back on in the opposite direction. Each of these tests were performed with the worst case possible for the network infrastructure, which implies 30 smart transducers streaming data at 1 kHz.

During the first test, when we shut down the power on one wireless router, we immediately lost 17 smart transducers. After 19 s, 10 were back online and it took a total of 32 s to recover the full network. After this recovery, the remaining router was at 72.6% of use. When we restart the previously stopped router, it took 5 minutes and 35 s to the network to stabilize (i.e no more smart transducer was switching to the other router). The second test is pretty straightforward. We removed the central unit (a Raspberry Pi 3) by removing its power source. We saw no change in the whole infrastructure at all but we lost the ability to change network configuration (i.e. drivers update). Finally, during the final test where we decided to shut down the entire house one breaker at a time, we gradually lost smart transducers since they don't have an autonomous power source until the last breaker switched. But the interesting result is the network came fully back to live when we have restarted everything, always with the one breaker at a time method.

These tests force our solution to show some reliability and self-healing capabilities. The first one, where we shut down one router, which can easily happen in a real house, demonstrates that the whole infrastructure was working again after a partial stop of only 32 s. This result is pretty impressive since we have removed half the infrastructure but it took only half a minute to fully come back online. The second test confirms us that the central unit is not mandatory for the whole architecture to work even if it could be the only Single Point of Failure. However it also shows us that, with the loss of this unit we could not add smart transducers anymore. To be fully reliable we have to cope with this problem. One solution could be to use more than one unit to introduce redundancy like in the network infrastructure with the two wireless routers. Finally, the last test proved us that even after a full power down, the architecture is able to fully restart by itself without any human intervention and since a power surge is a possibility in a house, it is a mandatory feature in such solution.

4.4 Pricing of our solution

To end this part, we have estimated the total cost of our solution in American dollars. The full pricing of our work is summed in Table 2. In this table, each line represent an item needed to build the infrastructure and the column correspond to the quantity needed, the unit price of the item and the total that is quantity by unit price. The exception to this rule is the last line of the table, which represents the total of every cost needed so the total of the whole architecture. As we can see, our whole solution costs barely 2000 USD. This is quite an improvement since the old one cost 13,500 USD without network infrastructure. Moreover, it should be noted that the Raspberry Pi Zero W price reported in this paper is the price of a full budget pack from Adafruit and not the price of the Raspberry itself, which is 5 dollars. We took the price of the pack over the real unit price because many items from the pack were used to perform this test but this price can be easily reduced if we bought only the elements needed (i.e. a SD card, the Raspberry Pi Zero W and a power source).

Since the vast majority of the aging population whose suffering from diseases or mental illness already struggle with many medical expenses (Bureau 2016; Alzheimer's Association 2017) consequently we have to keep our solution as cheap as possible. With a prototyping price below the 2000 US dollars we think this objective is partially achieved. Indeed, this price is lower than CASAS (2765 US dollars) and far lower than the LIARA and DOMUS architecture

Table 2 The total cost, without hardware sensors/transducers, of our infrastructure in USD

Item	Quantity	Unit price	Total
Wireless router	2	140	280
Pi zero W	30	35.50	1035
ESP32 thing	30	19.95	598.5
Pi 3	1	35	35
Total			1948.5

(13,500 US dollars) for more reliability, computing capabilities and transducers. But it does not contain the price of the sensing units (e.g. a thermometer, electromagnetic door sensor) that were present in the final pricing of the CASAS home (not in the LIARA and DOMUS architectures). However, these pieces of hardware a now relatively cheap and should not increase the whole price too much even if, we will have to take them into account if we want a full pricing of our solution. On the other hand, this price is also computed with only prototyping material not bought in large quantities. Consequently, we know that this number can be greatly improved with industrial manufacturing.

5 Conclusions and future works

In this paper, we presented a new kind of smart home architecture more reliable and scalable. Unlike the existing ones, our solution was designed not to have any Single Point of Failure, which can lead to reliability issues. We achieve such a result by adding intelligence and communication capabilities in every transducer in the house. To help the development of such infrastructure, we introduce a simple way to design and conceive these smart transducers. Thus, they are composed of two units, one charged to interpret raw sensors values and another one in charge of the intelligence and communication. These units communicate through a serial bus or an inter-process communication, which allows to retrieve high-level values and flash embedded drivers to ease the scalability of our infrastructure. Finally, the usage of Light Node Communication Framework, a messaging protocol defined in a previous work, improved the reliability in the whole infrastructure by removing the need of any central point for the communication.

We performed some tests on our new architecture to ensure the reliability and scalability capabilities as well as the performances. In terms of performances, we demonstrate that even in the worst case (i.e. 30 smart transducers sending raw values at 1 kHz) our infrastructure does not suffer from any latency except a 3 ms coming from the network. Moreover, this latency does not change when we moved from 5 to 30 smart transducers, confirming the scalability capabilities of our infrastructure. Regarding the reliability, we conduct a stress test by shutting down elements of the whole network and we showed that the proposed solution was able to recover in a relative short time after the loss of half or all the infrastructure. Moreover, we confirmed that the central unit we used to install the network and which could be considered as a Single Point of Failure was not mandatory for our network to work. Finally, we repost the price of our prototype to compare it with other architectures. With a price below 2000 US dollars, we can safely say that our solution is at a relative low-cost compared to other solutions which was an important goal.

Ultimately, we want to conduct more tests on our solution. First we want to remove any simulation from the tests by using only real smart transducers. This means as many hardware transducers as possible with real sensors or transducers and not random data generation and only real software transducer units (i.e. no random data generation either). This amelioration will help us prices the whole solution in real conditions. Then, once these improvements made, it should be interesting to put more stress on the architecture like increase the network overload to 100%, a number we never reached even at full capacity. Finally, this paper only describe an architecture for reliable and scalable smart homes but not how to provide services and activity recognition in a distributed manner on top of it. This blank must be addressed before any real life implementation.

References

Advantech (2016) Automation controllers and I/Os. http://www.advantech.com/products/automation-controllers-i-os/sub_1-2mlf31

Association Alzheimer's (2017) 2017 Alzheimer's disease facts and figures. Alzheimers Dement 13:325–373. https://doi.org/10.1016/j.jalz.2017.02.001. https://www.alz.org/documents_custom/2017-facts-and-figures.pdf

Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw

Augusto JC, Nugent CD (2006) Designing smart homes: the role of artificial intelligence, vol 4008. Springer, USA

Barker S, Mishra A, Irwin D, Cecchet E, Shenoy P, Albrecht J (2012) Smart*: an open data set and tools for enabling research in sustainable homes. ACM SustKDD

Bouchard K, Bouchard B, Bouzouane A (2012) Guidelines to efficient smart home design for rapid ai prototyping: a case study. PETRA

Bouchard K, Bouchard B, Bouzouanea A (2014) Practical guidelines to build smart homes: lessons learned. In: Opportunistic networking, smart home, smart city, smart systems (Book Chapter) pp 1–37

Bureau UC (2016) Household income. https://www.census.gov/data/tables/time-series/demo/income-poverty/cps-hinc/hinc-02.html

Callaway E, Gorday P, LH (2002) Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks. ieeexploreieeeorg

Chen M, Mao S, Liu Y (2014) Big data: a survey. Mobile Netw Appl 19(2):171–209

Chu-Sing Y, Mon-Yen L (2000) Realizing fault resilience in web-server cluster. In: ACM/IEEE SC 2000 Conference (SC'00), IEEE, pp 21–21, https://doi.org/10.1109/SC.2000.10012. http://ieeexplore.ieee.org/document/1592734/

Colombiano Kedowide EV Charles Gouin-Vallerand (2014) Recognizing blind spot check activity with car drivers based on decision tree classifier approach. AAAI Workshop—Technical Report WS, pp 22–26

Cook DJ, Youngblood M, Heierman E, Gopalratnam K, Rao S, Litvin A, Khawaja F (2003) MavHome: an agent-based smart home. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003 (PerCom 2003), pp 521–524, https://doi.org/10.1109/PERCOM.2003.1192783. http://ieeexplore.ieee.org/document/1192783/

Cook DJ, Crandall AS, Thomas BL, K NC (2012) CASAS: a smart home in a box. Computer 100(2):130–134. https://doi.org/10.1016/j.pestbp.2011.02.012.Investigations

Dell (2016) Dell PowerEdge rack servers. http://www.dell.com/ca/business/p/poweredge-rack-servers

Drumea A, Popescu C, Svasta P (2005) GSM solutions for low cost embedded systems for industrial control. In: 28th international spring seminar on electronics technology: meeting the challenges of electronics technology progress, 2005., IEEE, pp 240–244. https://doi.org/10.1109/ISSE.2005.1491034. http://ieeexplore.ieee.org/document/1491034/

Ghayvat H, Mukhopadhyay S, Gui X, Suryadevara N (2015) WSN- and IOT-based smart homes and their extension to smart buildings. Sensors (Switzerland) 15(5):10350–10379. https://doi.org/10.3390/s150510350

Giroux S, Leblanc T, Bouzouane A, Bouchard B, Pigot H, Bauchet J (2009) The praxis of cognitive assistance in smart homes. BMI Book pp 183–211. https://doi.org/10.3233/978-1-60750-048-3-183

Gutierrez J, Naeve M, Callaway E et al (2001) IEEE 802.15. 4: a developing standard for low-power low-cost wireless personal area networks. ieeexploreieeeorg

Hey A, Tansley S, Tolle K (2009) The fourth paradigm: data-intensive scientific discovery. Microsoft Research. http://202.120.81.220:81/inter/uploads/readings/four-paradigm.pdf

Hu Y, Tilke D, Adams T, Crandall AS, Cook DJ, Schmitter-Edgecombe M (2016) Smart home in a box: usability study for a large scale self-installation of smart home technologies. J Reliab Intell Environ 2(2):93–106

IEEE (1998) IEEE standard for a smart transducer interface for sensors and actuators

Intille SS, Larson K, Tapia EM, Beaudin JS, Kaushik P, Nawyn J, Rockinson R (2006) Using a live-in laboratory for ubiquitous computing research. In: International conference on pervasive computing. Springer, pp 349–365

King J, Jansen E (2005) The gator tech smart house. Computer 38(3):50–60

Lewis FL et al (2004) Smart environments: technologies, protocols, and applications. Wirel Sensor Netw 11:46

Lin RT, Hsu CS, Chun TY, Cheng ST (2008) OSGi-based smart home architecture for heterogeneous network. In: 2008 3rd international conference on sensing technology, IEEE, pp 527–532, https://doi.org/10.1109/ICSENST.2008.4757162. http://ieeexplore.ieee.org/document/4757162/

Liu B, Cao SG, He W (2011) Distributed data mining for e-business. Inf Technol Manag 12(2):67–79. http://link.springer.com/10.1007/s10799-011-0091-8

Lu F, Parkin S, Morgan G (2006) Load balancing for massively multiplayer online games. In: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games—NetGames '06. ACM Press, New York, p 1. https://doi.org/10.1145/1230040.1230064

Mon-Yen L, Chu-Sing Y (2001) Constructing zero-loss Web services. In: Proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), IEEE, vol 3, pp 1781–1790. https://doi.org/10.1109/INFCOM.2001.916676. http://ieeexplore.ieee.org/document/916676/

Montenegro G, Kushalnagar N, Hui J, Culler D (2007) Transmission of ipv6 packets over ieee 802.15. 4 networks. Tech. rep

Novák M, Binas M (2011) An architecture overview of the smart-home system based on OSGi. In: SCYR 2011: 11th Scientific Conference of Young Researchers of Faculty of Electrical Engineering and Informatics Technical University of Košice, pp 221–224

OSGi Alliance (2016) OSGi alliance the dynamic module system for java. https://www.osgi.org/

Patterson DJ, Liao L, Fox D, Kautz H (2003) Inferring high-level behavior from low-level sensors. In: International conference on ubiquitous computing

Plantevin V, Bouzouane A, Gaboury S (2017) The light node communication framework: a new way to communicate inside smart homes. Sensors 17(10):2397. https://doi.org/10.3390/S17102397. http://www.mdpi.com/231478

Postel J, Reynolds J (1985) File transfer protocol

Roy PC, Bouchard B, Bouzouane A, Giroux S (2013) Ambient activity recognition in smart environments for cognitive assistance. Int J Robot Appl Technol 1(1):29–56. https://doi.org/10.4018/ijrat.2013010103

Schroeder T, Goddard S, Ramamurthy B (2000) Scalable Web server clustering technologies. IEEE Netw 14(3):38–45. https://doi.org/10.1109/65.844499. http://ieeexplore.ieee.org/document/844499/

Shelby Z, Hartke K, Bormann C (2014) The constrained application protocol (CoAP). https://www.rfc-editor.org/info/rfc7252

ZeroMQ (2016) Zeromq. http://zeromq.org/

Zhihua S (2016) Design of smart home system based on ZigBee. In: 2016 international conference on robots and intelligent system (ICRIS), pp 167–170

Zou Z, Li KJ, Li R, Wu S (2011) Smart home system based on IPV6 and ZIGBEE technology. Proc Eng 15:1529–1533