Testing the Feasibility of Residential Wireless Interfaces Virtualization

Antonio da Silva Fariña,
Ana Belén García Hernando
Dept. of Telematic and Electronic Engineering
Universidad Politécnica de Madrid
Madrid, Spain
antonio.dasilva@upm.es,anabelen.garcia@upm.es

Mary Luz Mouronte López
Universidad Francisco de Vitoria
Madrid, Spain
maryluz.mouronte@ufv.es

***Abstract*—This work describes our proposal of leveraging two current networking trends, namely Network Functions Virtualization and Software Defined Networking, to enhance the flexibility with which residential IoT services can be offered to users. Concretely, we advocate for the potential virtualization of all the functionality that lies on top of the in-home wireless communications. These tasks, once virtualized, can be run in the cloud in the form of virtual machines or lightweight containers by using the Network Functions Virtualization capabilities of the Internet Service Provider. This way, the Internet Service Provider would also become a kind of "IoT Service Provider". From a practical perspective, we have chosen Universal Serial Bus as the boundary on top of which the functionality would be virtualized and remotely executed. The main reason for this election is that Universal Serial Bus is widely adopted as a residential interface and there is an ample variety of wireless communications that can be accessed through Universal Serial Bus dongles with the appropriate drivers. Our idea is to tunnel the Universal Serial Bus blocks between the low level IoT gateway at home (which we have called Home Radio Head, or HRH) and the Internet Service Provider in order for the latter to access these flows and treat them accordingly to build IoT services for the residential user. These tunnels could be dynamically provisioned by using Software Defined Networking protocols. In this paper we elaborate not only on our architecture but also on the results we have obtained from tests run on a pilot, considering several networking configurations between the residential environment and the emulated Internet Service Provider premises. These tests range from purely functional proof of concepts to estimations of bandwidth and delays incurred when using our proposal. The figures obtained show that the idea is feasible, and point to cases where executing part of the virtualized functionality inside the Home Radio Head might be necessary for performance or bandwidth usage reasons.**

Keywords-Residential IoT; Network Functions Virtualization; Software Defined Networking; USB interfaces virtualization; USB/IP tunneling.

I. INTRODUCTION

This paper is an extension of a previous one, of the same authors, accepted and presented during the Thirteenth International Conference on Wireless and Mobile Communication (ICWMC 2017) [1]. In that work we introduced our proposal for an architecture that leverages two current networking trends, namely Network Functions Virtualization (NFV) [2][3] and Software Defined Networking (SDN) [4], to augment the flexibility when offering a portfolio of IoT services to residential users. This proposal consists in the virtualization and remote execution of the functionality that lies on top of the residential IoT wireless communications. This way, different IoT manufacturers' hardware can be more easily supported, since the specific drivers and all the upper level tasks can be run by the Services Provider and even activated in a "plug and play" manner for the users.

By residential IoT we mean the part of the Internet of Things that is or will be deployed mainly for private homes or small companies (i.e., SOHO - Small Office Home Office). These environments do not usually pose such strict constraints regarding availability, safety or tolerance to harsh conditions as industrial, environmental, healthcare or outdoor scenarios do (to name a few). In [5], different IoT application domains are compared in terms of traffic characteristics and QoS requirements; in the case of "smart buildings and living", there are small network sizes (tens or at most hundredths of nodes) and short coverage areas, the traffic rate is low (or medium at most), and the tolerance to delay usually ranges between seconds to tenths of seconds.

However, the high capillarity of homes with Internet access that will deploy IoT services, and the ample variety of proprietary solutions on the market, make it very convenient to be capable of offering a unified and virtualized support for all the residential IoT solutions.

Nowadays, application-layer gateways are usually needed to provide connectivity to IoT devices in the home. Current gateways mix network connectivity, in-network processing, and user interface functions. We share the view of [6] by which separating these functions would improve the connectivity potential for IoT devices. In fact, there are different publications that consider NFV and SDN as two elements of the new ISPs (Internet Service Providers) architecture to flexibly support IoT [7].

Our specific approach requires for the low level flows of information to be tunneled and consumed at the ISP side, and we opted for Universal Serial Bus (USB) as the boundary between the purely wireless in-home communication and the raw information that is to be processed remotely. This way, a generic and programmable gateway would be present at home, called Home Radio Head (HRH) (a name inspired on the Remote Radio Heads (RRH) present in some cellular network deployments), which does not need to implement any IoT vendor-specific function above the wireless communications provided by

USB dongles. USB interfaces would be virtualized and managed remotely thanks to the establishment of tunnels between the HRH and the ISP.

At the remote end of these tunnels, a NFV infrastructure hosts the IoT applications and management functionality implemented as a set of Virtual Network Functions (VNFs). The virtualization of USB interfaces allows to reach economies of scale by offering a reasonably inexpensive customer premises equipment supporting a wide variety of home wireless communications.

Moreover, we propose to also introduce the programmability of SDN into our architecture, so that the USB tunnels are dynamically provisioned by an SDN controller that communicates with the HRH through a southbound interface such as OpenFlow [8].

In order to reduce the necessary bandwidth between home and the ISP, or if the delay/latency requirements are very stringent for a specific application, some processing can be carried out inside the customer premises by downloading and running lightweight Virtual Machine (VM) containers [9] in the HRH.

In our paper [1] we also showed our first results on the feasibility of this approach, consisting on estimations of the bandwidth attainable through USB/IP tunneling [10] between the home and the ISP under different networking scenarios.

This paper elaborates more on the feasibility and quantitative study of our proposal. Firstly, an actual low rate video flow generated by a USB WebCam has been successfully transported and visualized by a virtual function located in an emulated ISP. Secondly, round trip time (RTT) measurements are taken and compared for local and remote (virtualized) scenarios. This gives the idea of when it is necessary to execute some of the virtualized functionality inside the HRH, which, besides being SDN manageable, could incorporate some kind of lightweight virtualization capacities as indicated above.

The rest of the paper is organized as follows: Section II summarizes the background and some related work. Section III describes our proposed architecture and its building blocks. Section IV explains the experimental setup we have implemented and discusses the results. Finally, in Section V, we provide the conclusion and our foreseen future work.

II. Background

In this section we firstly review the work related to the use of NFV and SDN in home environments. Then we elaborate on the concept of the virtualization and remote execution of radio functionality, which is used in current cellular networks and inspires our proposal for a HRH. After that, the main lightweight virtualization options that could be useful for executing part of the VNFs in the HRH are described. Finally we summarize the current literature on the virtualization of residential IoT.

A. NFV and SDN in Home Environments

NFV leverages commodity storage, networking and processing equipment in order to execute, through the use of a virtualization layer (sometimes called hypervisor), sophisticated network functionality on top of a virtualized infrastructure. It may be used to combine the available resources in a network by dividing the bandwidth into channels or slices, each of which is independent from the others. NFV allows multiple service providers to construct different separate and isolated virtual networks, which share physical resources.

The standardization of NFV started with ETSI, and several use cases have been defined in [11], such as the virtualization of the home environment and the virtualization of Internet of Things.

With SDN, the control plane (in which the logical procedures supporting the networking protocols and the most important decisions are made) is separated from the data plane (in which the forwarding of packets on the most suitable interface towards the intended destination is carried out). SDN is an excellent mechanism to do Traffic Engineering (TE) and exploit effectively the network resources in an IoT scenario.

These two technologies, far from being incompatible, are increasingly being considered together in new architectures. The main use cases of SDN and NFV in the home deal with pure networking tasks, and more specifically with the virtualization of the Customer Premises Equipment (CPE). There are also some recent proposals to augment the scope of cloud computing, NFV and SDN and integrate some IoT (basically sensing and actuation) capabilities into their frameworks [12][13].

In some use cases [14], hardware middleboxes are deployed by cloud providers, executing several network functions, and enhancing the cloud capability. To solve the costs, manageability, and performance overhead problems, NFV has been proposed as a good solution and therefore, software applications have been deployed in place of the hardware middleboxes. The high computing power brought by NFV would provide rich-media functionalities to thin customer devices and would change the way the multimedia services and applications are used.

The paper [6] proposes an architecture that leverages the increasingly ubiquitous presence of Bluetooth Low Energy radios to connect IoT devices to the Internet; several example applications are shown, and several research challenges in its implementation are investigated.

Home environments where the hardware resources are shared efficiently and overviews of different virtualization mechanisms to cooperate among home networks have been investigated, comparing methods and concepts [15]. Key technical challenges behind this idea, such as dynamic allocation, migration, and orchestration of virtual machines across wide areas of interconnected edge networks have been analyzed [16], and new mechanisms to allow that residential users control the access network resources and manage different types of traffic to fulfil the Quality of Service (QoS) requirements have been proposed [17]. Some studies on how to place NFV to satisfy the QoS requirements have also been made [18][19].

The work [20] describes an efficient network management proposal by means of home gateways that focus on supervising the network traffic flows, executing a per-flow management, and implementing a custom DHCP (Dynamic Host Configuration Protocol) to enable traffic segregation and its proper measurement at the IP (Internet Protocol) layer. The authors of [21] discuss novel solutions to build a virtual residential gateway using an SDN controller at the service provider side to manage services in home. Another research on residential gateways that can be controlled and managed remotely via an SDN controller, adjusting and troubleshooting the residential network, can be found in [22].

No doubt, security is an important issue when it comes to controlling and managing lots of in-home residential equipment. The security challenges in SDN/NFV [23] and the feasibility of extending the current NFV orchestrator to provide security mechanisms have been recently studied [24]. The proposed security solutions supervise parameters, generate access control policies, and apply them through the underlying infrastructure. Most solutions can work together with the NFV orchestrator to enable fine-grained access management to safeguard services and resources.

The novelty of our proposal lies in the fact that we virtualize all the functionality above the very low level communications between in-home IoT devices and the HRH. This augments the flexibility and can even lead to a real "plug and play" support of new hardware on the part of the ISP, as long as there is a USB dongle plugged in the HRH with which the sensors and actuators communicate.

Our architecture can be seen as one possible realization of a specific instance of the use case "Virtualization of Internet of Things (IoT)" presented in [11], including as an additional element the use of SDN.

The existing literature on the usage of NFV and/or SDN for home environments is usually dependent on the existence of an in-home hub or concentrator (sometimes called Customer Premises Equipment). Many of these proposals include the virtualization of part or all the functionality of this element, whereas ours is, to the best of our knowledge, the first that advocates for the virtualization of all the vendor-specific IoT functionality above a very low-level interface (USB in our case), moving everything else to the ISP. This substitutes the IoT gateway that is maintained in many proposals by a generic HRH easily reusable for new IoT products.

We stress here that our proposal does not prevent the existence of higher-level functions such as service publication and recognition, data analysis or application-specific processing. These functions can be present above the virtual USB points of presence at the NFV infrastructure on the ISP side, in the form of one or several VNFs.

B. *Virtualization and Remote Execution of Radio Functionality*

Our HRH is inspired by the Remote Radio Head (RRH) approach used in cellular wireless access networks, which aims to move wireless baseband processing to the cloud. This approach has a high cost in terms of bandwidth that is solved using dedicated high-speed lines connecting the RRH with the Base Band Unit (BBU) at the edge of the core network.

Recent researches exist on RRH, such as [25], which describes the process of moving from RRH scenarios where the BBU are deployed at the base of the cell tower, to an architecture in which the BBU are located in several centralized locations. The authors of [26] present an overview of the state of the art on Cloud Radio Access Network (C-RAN) research, with focus on front-haul compression, baseband processing, medium access management, resource allocation, system-level requirements and standardization works.

We find that the virtualization of higher level functionality is appealing not only for cellular networks but also for residential environments. However, the specificities of local area protocols, inherently different from cellular wireless, make it necessary to assess to what extent and under what circumstances this externalization of functions is feasible. In this paper we provide some results obtained from tests that range from the feasibility of consuming actual traffic generated by a WebCam to the measurement of round trip time delays under different networking scenarios between the home and the ISP.

C. *Lightweight Virtualization Environments*

The orchestration and maintenance of the software running on the gateways in large-scale deployments is a challenging task. There are studies, such as [27], that evaluate the performance of the container-based approach compared to a hypervisor-based virtualization when running on gateway devices. The comparison between traditional heavy VMs, Unikernels and Docker containers is shown in Fig. 1. Unikernels [28] are specialized single-purpose operating systems, which are several magnitudes smaller than general-purpose OS (Operating Systems); they can also be used in small Internet-of-Things (IoT) devices intended to execute a specific software application. On their side, Docker containers are a type of lightweight VM that can be easily deployed in inexpensive common single board computers like Raspberry Pi. [27][29][30] also conclude that Docker presents better performance than the traditional VM as it has no guest operating system and its overhead is considerably reduced. The lightweight virtualization has been also shown to be a suitable technique for auditing applications, detecting intrusions, and recovering systems from attacks and errors [31].

Ultimately the level of security guaranteed by applications developed within containers has become an important concern [32]. The latest versions of Docker have included several security enhancements to solve several security issues. Even more, Docker team continuously publishes guidelines in order to avoid security threats and build safer Docker ecosystems [32][33]. Additionally, a developer's tool that allows examining many security issues within virtualized applications has been implemented as a collaboration work between Docker and the Center for Internet Security [34].

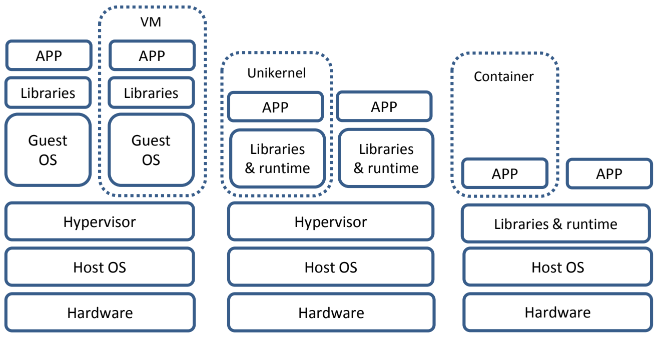


Figure 1. Comparison of traditional VM (left), Unikernels (center) and Container (right) virtualization architectures.

In any of its forms, this lightweight Platform-as-a-Service (PaaS) environment is very useful to deploy custom home sensors preprocessing functionality at runtime. With it, both the necessary bandwidth to the cloud and the round trip delay for those functions can be reduced.

The possibility of distributing the resource-intensive functions in intermediate points between the end devices (e.g., sensors and actuators) and the cloud is aligned with the philosophy behind fog computing approaches. This distribution has to be done transparently for the users, and a certain amount of intelligence is needed to manage a fog scenario, preferably in an open and interoperable manner. IoT and 5G [35] are two of the drivers that currently push the activity in fog computing.

Recent researches [35] describe novel IoT architectures based on SDN and Fog computing, where SDN provides a centralized network control plane, which executes complex mechanisms for traffic and resource management, and fog computing allows that much information is processed and managed at the network edge, being able to support applications with very low latency requirements. Also related to this, in [36] the authors review the distributed approach to NFV, discuss phased NFV deployment and present critical factors to take such functionalities into account at the customer edge.

Our proposal is in a way aligned and compatible with fog/edge computing paradigms, in the sense that we do not restrain the locations at which the IoT specific functionality has to be run. This depends on the NFV infrastructure that the ISP has, and on the policies in place to decide how to split the different VNFs that form a specific service. Functions that are more time-sensitive (e.g., reactions to events that raise alarms) can be executed near (or at) the network edge or even inside the HRH, whereas more delay tolerant tasks (e.g., HVAC -Heating, Ventilation, & Air Conditioning- activation due to temperature changes or statistics collection) can be run in more centralized locations to be shared by more customers.

D. Virtualization of Residential Internet of Things

Building functions that cope with all the diversity that the home IoT products present is not feasible, at least currently and at a reasonable cost, for the residential user. However, for an Internet Service Provider (ISP) this would be much easier, especially if these functions are offered as services to its customers and economies of scale can be applied. The ISP should virtualize the actual physical infrastructure of its customers to deliver a set of general and reusable services [37]. In fact, sensing as a service (S2aaS) architectural proposals are specifically concerned with the organizational relationships between the different components and omit details about short range components communications as well as other technical aspects [38].

This is a very active research field. Among the recent works in this area we highlight the following. In [39], the authors survey the state of the art on the application of SDN and NFV to IoT. They provide a description of the possible implementation aspects for both technologies.

The work [12] highlights some IoT challenges that the network and the IT (Information Technologies) infrastructure will face. The NFV and SDN benefits are presented from the point of view of the network operator. The authors present a new multi-layered IoT architecture involving SDN and NFV, and they show how the proposed architecture is able to cope with the identified IoT challenges.

In [13], the authors discuss the usage of NFV technologies and construct a virtual advanced metering infrastructure (AMI) network to transmit energy-related information in a dependable and cost-effective way. The reliability, availability and cost of the new architecture is analyzed and compared to current AMIs. Another example of a specific application is found in [40], in which the authors propose a Logical Access Point-based Mobility Management (LAPM) scheme for WLAN (Wireless Local Area Network), based on an extended SDN/NFV abstraction, which outsources the IEEE 802.11 protocol stack complexity to a centralized controller.

In [6], an architecture that leverages the increasingly ubiquitous presence of Bluetooth Low Energy radios to connect IoT peripherals to the Internet is proposed. The authors propose the use of mobile devices (i.e., Laptops, Smartphones and tablets) as gateways. The same approach is followed in [41], where the use of smartphones running as gateway bridges with Bluetooth-enabled devices in a home environment is evaluated.

We finally mention the work [42], in which a new user-centric management architecture is proposed, to increase the active engagement of residential users in the management tasks of their own networks, improving the usability of the network and facilitating the provision of new services. The proposed architecture combines the SDN and NFV approaches. Additionally, the user-centricity is achieved by implementing interaction and management layers. These layers together constitute a residential network management application. The interaction layer, which can be deployed over different devices, hosts the application that allows the user to configure the network and receive notifications. The interaction layer interacts with the management layer by means of a REST API (Representational State Transfer Application Programming Interface).

III. Proposed Architecture

In our previous work [37], we described our proposal for an architecture that leverages NFV and SDN to offer a portfolio of IoT services to residential users.

The virtualization of IoT vendor-specific functionality, together with the presence of a cost-effective and generic customer premises equipment called HRH, would bring economies of scale, easier updates and faster support for new IoT products, among others. This in-home HRH:

- Would not need to implement IoT vendor-specific functions, since these functions would be virtualized and run on a standard NFV infrastructure.
- Would send the in-home raw layer 1 physical flows to the ISP by using tunnels.
- Would be SDN-manageable to establish and maintain the aforementioned tunnels in a standardized manner.

With the work described in this paper we elaborate further on this architecture and decide to establish USB as the vendor-agnostic frontier between the in-home radio flows and the vendor-specific functionality that would be implemented as VNFs. In Fig. 2 we show the high-level view of our proposal.

Our architecture is based on the following principles:

- The support of varied short-range wireless interfaces widely used by residential IoT products (e.g., WiFi, Bluetooth, ZWAVE or ZigBee). These radio flows are processed locally at low level and exposed to the HRH as standard USB interfaces.
- The establishment of tunnels between the HRH and the virtualization infrastructure at the ISP side, in order to propagate transparently the USB Request Blocks (URBs) that are to be processed by VNFs. The establishment of these tunnels is programmable by using SDN, i.e., the HRH supports Openflow (or any other SDN-compliant southbound interface).
- The vendor-specific functionality, which is realized by means of one or several VNFs (represented as "vHOME" virtual machines in Fig. 2), can be distributed if necessary. In this line, the HRH supports a lightweight virtualization environment on which a subset of the VNFs necessary for a service can be downloaded and executed, as commanded by the Virtualized Infrastructure Manager (VIM), see Fig. 3.

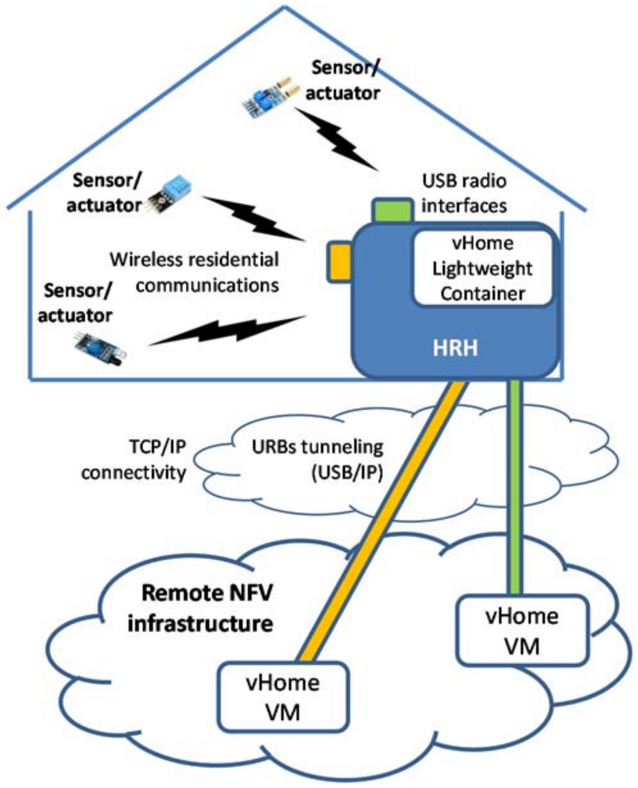


Figure 2. High-level architecture for the remote virtualization of wireless residential communications by means of USB tunneling.

Some of these architectural elements were already drafted in our work [37], such as the use of NFV and SDN, the support of different short-range wireless communications in home, the virtualization of higher-level functionality at the ISP and the possibility of having a lightweight virtualization environment in the HRH. What our current proposal advances with respect to [37] is a much clearer and concrete boundary on what is executed in home and what is virtualized at the ISP, by the selection of USB as the technology whose blocks are going to be tunneled, instead of considering generic "raw layer-1 flows" either at bit or frame level. This way, the "virtual NICs" (virtual Network Interface Cards) that were present in [37] become "vUSBs" in Fig. 3, and the HRH becomes a device with the possibility of having several USB dongles plugged in (to support different wireless or wired IoT products), instead of having to embed the different wireless technologies itself.

Even if establishing USB as the boundary between the in-home and the ISP functionalities can be considered as a limitation to our more general initial architecture, the fact that it is a widely used interface for residential consumer electronics clearly brings advantages with respect to the feasibility of deploying our solution in a wide scale. The fact that the lowest level wireless functions are going to be embedded into the USB dongle is also convenient, since the first establishment of a flow by SDN procedures can take significantly longer than subsequent transmissions, and that initial delay might hinder the beginning of the wireless interactions. Moreover, the HRH becomes a more general device since USB is a well-known technology, and the support of additional wireless technologies would only require to have the adequate USB dongle, acquired together with the rest of the IoT hardware from the vendor. This technological and architectural decision has also allowed us to prove the feasibility of our proposal, as will be explained later in the paper.

In the following sub-sections, we elaborate further on the implications, advantages and rationale behind each of our main architectural design decisions.

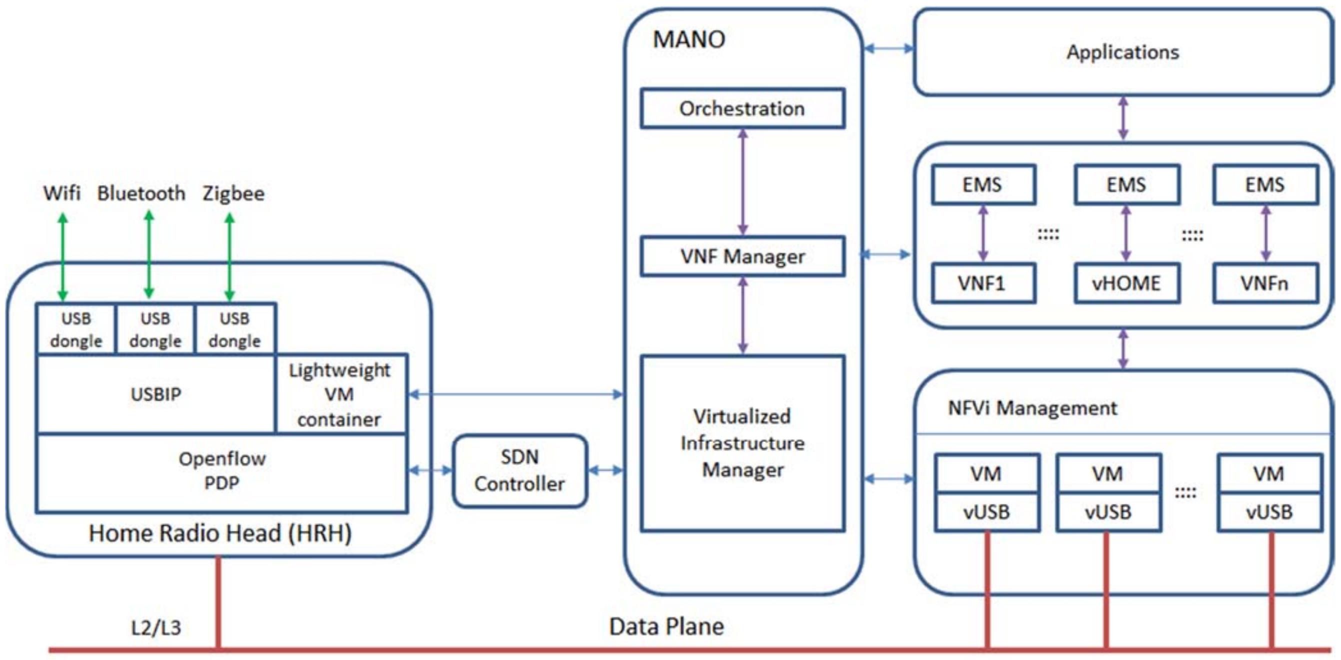


Figure 3. SDN and NFV as considered in our proposal.

A. *Virtualization of USB Interfaces*

USB Redirection consists of plugging an external device into a USB port on a local endpoint and accessing that device from within a remote system or application. The rationale behind the virtualization of USB interfaces that we include in our proposal is threefold. Firstly, USB is widely supported by IoT vendors in residential environments. Secondly, the existence of USB dongles at reasonable prices and easily interchangeable is very convenient for the residential market. Lastly, fast prototyping of proof of concepts becomes possible with general purpose equipment (see Section IV below).

Moreover, virtualizing the functionality above the USB interface would make various existing IoT products immediately available through our proposed schema. New products also supporting USB would become available to customers almost in a “plug-and-play” manner as long as ISPs supported the adequate virtualized drivers.

Each tunnel would correspond to a new IoT product that utilizes a specific wireless technology (see Fig. 2), and the specific drivers and all upper functionality would be placed at the other end of each tunnel, on the ISP side. To implement this idea, we propose to use USB/IP [10], a means of sharing USB devices over a TCP/IP (Transmission Control Protocol / Internet Protocol) network by encapsulating USB messages between a server (the equipment with the USB device physically connected) and a remote client.

B. *SDN-Programmable Data Path*

In order to provide a flexible configuration, the URB flows should be dynamically provisioned and managed. The HRH would benefit from a generic datapath that is programmable by following the SDN principles. The concrete policies to be applied to the establishment of the tunnels would be implemented and enforced by an SDN controller, and a southbound Openflow-programmable datapath has to be supported by the HRH. This way, SDN advantages brought by the software definition of networking configuration are present in our scenario. Also, our HRH would be more easily integrated with an SDN-based residential gateway as proposed in [42].

C. *Lightweight Docker Containers*

Under certain circumstances, it might be convenient or even necessary that a subset of the vendor-specific functionality is run inside the customer premises. This might be the case for complying with stringent delay requirements or for saving uplink/downlink bandwidth. We propose to provide a light virtualization environment, based either on light virtual machines or on Docker containers, inside the HRH, in which specific modules can be downloaded and executed locally when commanded by the NFV management and orchestration layer.

This distribution of functionality has to be done transparently, without the user being aware of the decomposition of the global service into different modules that may be executed at different points.

D. *SDN/NFV Relationship*

Our HRH follows the principles of both SDN and NFV. As such, it contains on one hand a generic and programmable networking datapath, and on the other hand a lightweight virtualization environment. The former offers a standard SDN southbound interface so that the SDN controller can provision the USB tunnels dynamically. The latter is formally part of the virtualized infrastructure that has to be managed by the VIM, as per the NFV architecture.

Fig. 3 is an enhanced version of a figure we included in our previous paper [37]. We have completed the modules and technologies inside HRH, and we have also made the virtualization capabilities of HRH explicit. The SDN controller that is functionally located between the VIM and the HRH (see Fig. 3) can itself be implemented as another VNF, this way leveraging the existing NFV infrastructure.

IV. Experimental Setups and Results

In order to validate our approach, we have carried out several experimental setups. To better assess the reproducibility of these experiments, we highlight here that all the hardware used is inexpensive and off-the-shelf, and all the software is open source. In the same way, and to guarantee the generality of the tests carried out, the test benches are implemented using both Windows and Linux as guest operating systems.

A. *USB Redirection Tests*

We have implemented the experimental setup shown in Fig. 4. To act as HRH, we have equipped a Raspberry Pi 3 with a USB/IP server running on Raspbian OS. This HRH is located inside the Smart Home that the Universidad Politécnica de Madrid has in its South Campus. Both a generic USB WebCam and a USB mass storage device (i.e., a USB pendrive) are connected to the HRH.

The ISP side is emulated by means of a Windows PC equipped with a VirtualBox hypervisor. On top of this virtualization infrastructure, a guest OS is run that contains a USB/IP client. This client is in charge of terminating the USB tunnels and offering the virtualized USB dongles to the guest OS as if they were local. On top of these virtualized USB dongles, the specific functionality can be deployed.

We have designed and executed two USB redirection tests. The first one is actually composed of different measurements. The objective of this composed test is to estimate the bandwidth that would be available through the USB tunneling infrastructure for different local-remote networking scenarios. To perform this estimation, we have run several write and read tests on the regular USB pendrive and have measured the performance of those operations. Four local-remote setups have been considered:

- Local: This is the baseline that gives us the actual write/read capacity of the USB device. Both operations are performed on a pendrive connected to the same node.
- Same network: The HRH and the emulated ISP are connected to the same Ethernet network. In our case, it is a 100BASE-TX wired Ethernet connection.
- Madrid-Alcalá: The HRH is located inside the Smart Home in the South Campus of UPM (Universidad Politécnica de Madrid), whereas the emulated ISP is connected to a residential network in Alcalá de Henares, a village on the outskirts of Madrid. They are around 30 km. apart from each other. The residential network has a 50 MB/s Hybrid fiber-coaxial (HFC) internet connection.
- Madrid-Galicia: The HRH is located inside the Smart Home in the South Campus of UPM, whereas the emulated ISP is connected through a WIFI access point in a public library in Galicia (North of Spain). They are around 500 km. apart from each other. The internet connection of the library, which is provided by a public ISP, is a 100 MB/s HFC access shared by all users.

These setups are chosen to consider scenarios in which not only the distance but also the expected quality of the networking accesses is varied. We have compiled the obtained results in Table I below.

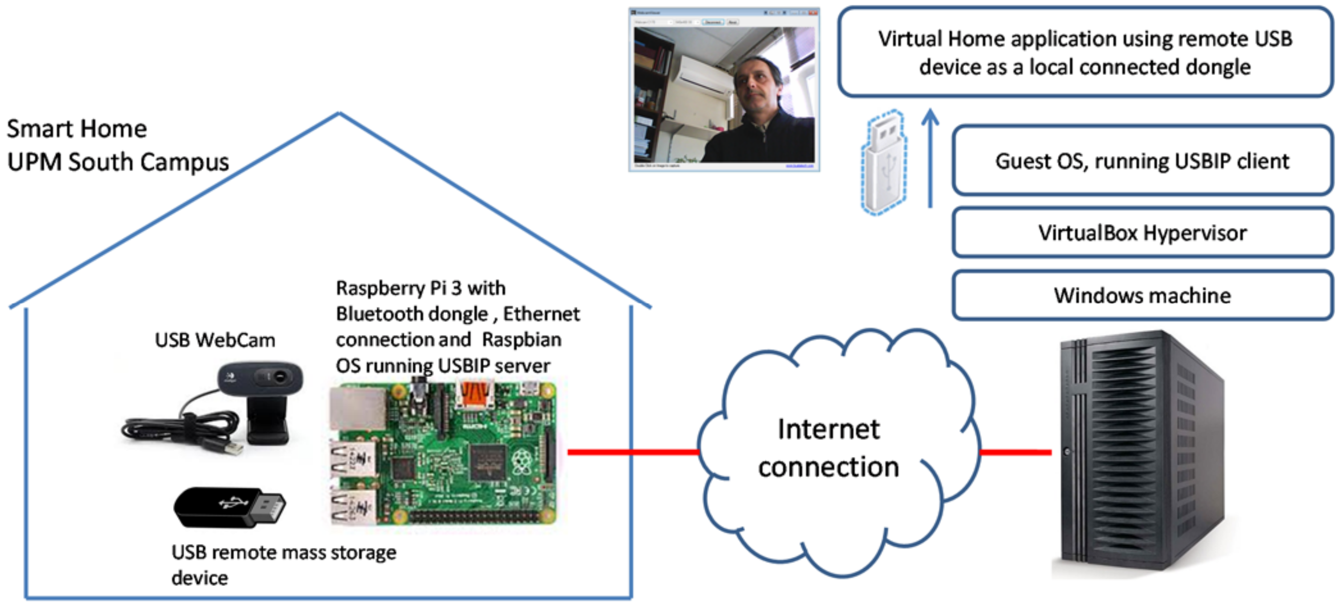

Virtual Home application using remote USB device as a local connected dongle
Guest OS, running USBIP client
VirtualBox Hypervisor
Windows machine
Smart Home
UPM South Campus
Raspberry Pi 3 with Bluetooth dongle , Ethernet connection and Raspbian OS running USBIP server
USB WebCam
USB remote mass storage device
Internet connection

Figure 4. USB redirection tests setup.

TABLE I. WRITE / READ PERFORMANCE UNDER SEVERAL LOCAL-REMOTE CONFIGURATIONS

Configuration	Write	Read
Local	7.34 MB/s	17.5 MB/s
Same network	4.32 MB/s	6 MB/s
Madrid-Alcalá	0.9 MB/s	1 MB/s
Madrid-Galicia	0.4 MB/s	0.5 MB/s

We have specified the concrete type of Internet access that is present at each setup location to give a better idea of the influence that this might have on the final perceived figures included in Table I. Even though we are aware that different specific locations would have thrown different numbers, we consider that the objective of demonstrating the feasibility of virtualizing and executing remotely many of the usual residential functions is reasonably well attained.

Anyway, these bandwidth measures should be taken as a starting point, because they have been obtained in controlled environments that implement per-user traffic shaping or bandwidth limits policies to control peer to peer communications. Nonetheless, even in the most disadvantageous scenario, the bandwidth estimations show that for sporadic or periodical sensor readings (such as temperature or humidity) and for short actuator orders (such as lights on/off control), it is feasible to execute all virtualized functions remotely. In fact, most smart home sensing-and-actuation applications are supposed to present low rates to the network, since each sensor will usually produce one reading (one message) every several minutes [5]. Even in the case of having tenths or near one hundred sensors, it is clear that the traffic rate produced by these applications is compatible with the measurements we have obtained.

Even low-rate video, which would generate a significantly higher throughput than sensor and actuator readings and orders, can be remotely tunneled if necessary: as an example a Youtube video with a resolution of 360p consumes around 0.3/0.4 MB/s. However, in the case of more intensive multimedia traffic, such as a video camera output with higher resolution, it might be necessary to download and execute some of the processing functionality into the HRH, in order to consume less bandwidth towards the ISP. This decision can be made on the basis of bandwidth or delay measurements with each residential subscriber, which could be easily taken by the ISP.

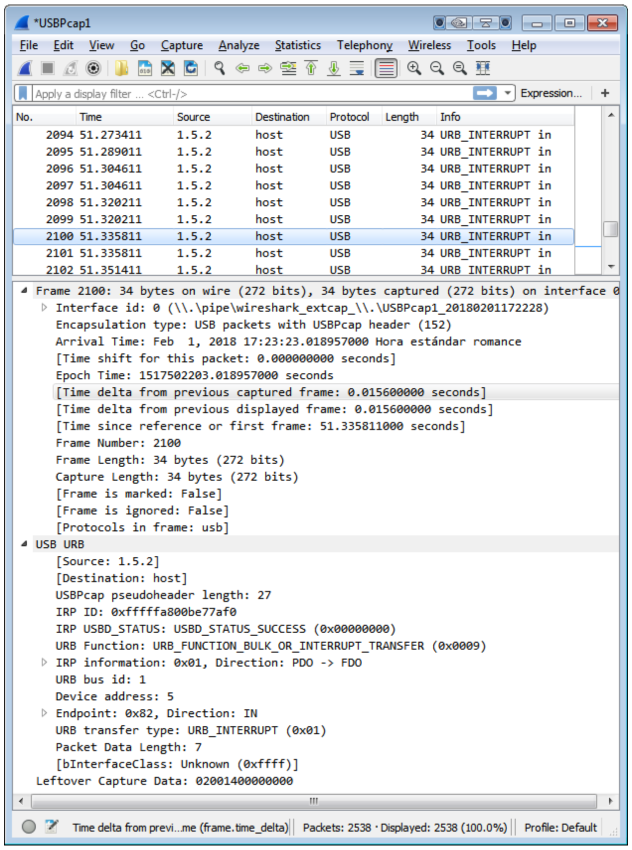
The second USB redirection test evaluates the capability of tunneling periodic USB traffic and processing it in a virtualized environment. To do this, we have used a conventional USB WebCam.

There exist different types of USB transfers, e.g., isochronous transfers are meant for transmitting real-time information such as audio and video, and must be sent at a constant rate. USB isochronous data streams are allocated a dedicated portion of USB bandwidth to ensure that data can be delivered at the desired rate. An isochronous pipe sends a new data packet in every frame, regardless of whether the delivery of the previous packet was successful or not.

On the other hand, interrupt transfers are intended for devices that send and receive data asynchronously. The interrupt transfer type guarantees a maximum service period and that delivery will be re-attempted in the next period if an error occurs on the bus. This transfer protocol is ideal for time sensitive applications because it has a guaranteed bound latency. A typical example is a USB mouse peripheral.

To analyze closely the local USB flow generated by the WebCam we have used Wireshark. Wireshark is a widely used network protocol sniffer, which can also be used to capture raw USB traffic on local endpoints. Fig. 5 shows the result of capturing the traffic in the local WebCam USB connection. As can be seen, in this case the device uses an interrupt transfer mode (even if the isochronous type may seem more adequate). The bandwidth required by the WebCam is not high since the data packets are seven bytes long.

Once this is known, a purely functional test was made. The USB WebCam was plugged in a local endpoint of the raspberry board and the traffic was transmitted using USB/IP to the ISP side. This way, the device was made available to the VM running in the cloud (in the emulated ISP). An off-the-shelf open source WebCam Viewer was used to open the device and play the remote video flow as if it was a local WebCam. This setup worked well and the video was correctly visualized.


*USBPcap1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.        Time       Source    Destination  Protocol  Length  Info
    2094 51.273411    1.5.2     host         USB          34 URB_INTERRUPT in
    2095 51.289011    1.5.2     host         USB          34 URB_INTERRUPT in
    2096 51.304611    1.5.2     host         USB          34 URB_INTERRUPT in
    2097 51.304611    1.5.2     host         USB          34 URB_INTERRUPT in
    2098 51.320211    1.5.2     host         USB          34 URB_INTERRUPT in
    2099 51.320211    1.5.2     host         USB          34 URB_INTERRUPT in
    2100 51.335811    1.5.2     host         USB          34 URB_INTERRUPT in
    2101 51.335811    1.5.2     host         USB          34 URB_INTERRUPT in
    2102 51.351411    1.5.2     host         USB          34 URB_INTERRUPT in
```

```
Frame 2100: 34 bytes on wire (272 bits), 34 bytes captured (272 bits) on interface 0
   Interface id: 0 (\\.\pipe\wireshark_extcap_\\.\USBPcap1_20180201172228)
   Encapsulation type: USB packets with USBPcap header (152)
   Arrival Time: Feb  1, 2018 17:23:23.018957000 Hora estándar romance
   [Time shift for this packet: 0.000000000 seconds]
   Epoch Time: 1517502203.018957000 seconds
   [Time delta from previous captured frame: 0.015600000 seconds]
   [Time delta from previous displayed frame: 0.015600000 seconds]
   [Time since reference or first frame: 51.335811000 seconds]
   Frame Number: 2100
   Frame Length: 34 bytes (272 bits)
   Capture Length: 34 bytes (272 bits)
   [Frame is marked: False]
   [Frame is ignored: False]
   [Protocols in frame: usb]
USB URB
   [Source: 1.5.2]
   [Destination: host]
   USBPcap pseudoheader length: 27
   IRP ID: 0xffffa800be77af0
   IRP USBD_STATUS: USBD_STATUS_SUCCESS (0x00000000)
   URB Function: URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x0009)
   IRP information: 0x01, Direction: PDO -> FDO
   URB bus id: 1
   Device address: 5
   Endpoint: 0x82, Direction: IN
   URB transfer type: URB_INTERRUPT (0x01)
   Packet Data Length: 7
   [bInterfaceClass: Unknown (0xffff)]
Leftover Capture Data: 02001400000000
```

```
Time delta from previ...me (frame.time_delta) | Packets: 2538 · Displayed: 2538 (100.0%) | Profile: Default
```

Figure 5. Local USB WebCam data sniffing with Wireshark.

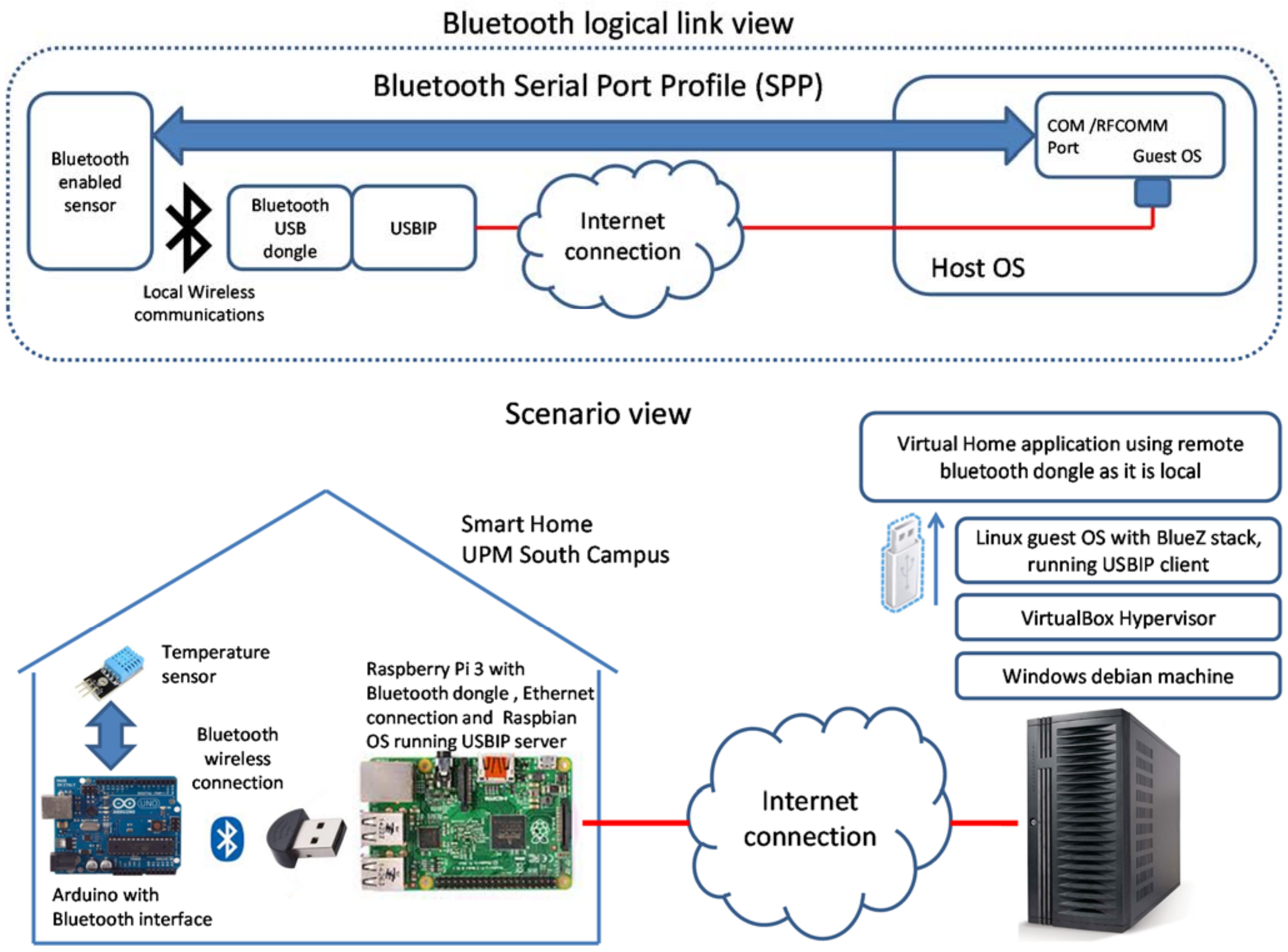

Bluetooth logical link view
Bluetooth Serial Port Profile (SPP)
Bluetooth enabled sensor
Bluetooth USB dongle
USBIP
Internet connection
COM /RFCOMM Port
Guest OS
Host OS
Local Wireless communications
Scenario view
Virtual Home application using remote bluetooth dongle as it is local
Linux guest OS with BlueZ stack, running USBIP client
VirtualBox Hypervisor
Windows debian machine
Smart Home UPM South Campus
Temperature sensor
Bluetooth wireless connection
Raspberry Pi 3 with Bluetooth dongle , Ethernet connection and Raspbian OS running USBIP server
Arduino with Bluetooth interface
Internet connection

Figure 6. Bluetooth redirection tests setup.

B. *Local Bluetooth Redirection Tests*

The tests described above only show that it is feasible to use remote USB devices as if they were locally connected. We went further with our next experiments, to prove that it is possible to use a Bluetooth USB dongle and have the wireless interface remotely accessible.

To do this we have deployed the setup shown in Fig. 6. The ISP side is emulated by means of a Windows PC equipped with a VirtualBox hypervisor. On top of this virtualization infrastructure, a guest Linux OS runs a USB/IP client interface and the BlueZ Bluetooth stack. All the communications between the sensors and the cloud virtual home control application are Bluetooth.

Any BT (Bluetooth) device must be compatible with at least a subset of the profiles defined by the Bluetooth specification. For example, hands-free systems located in cars have to comply with the Cordless Telephony Profile (CTP) in order to communicate with mobile phones. In our case, the Serial Port Profile (SPP) is used. It emulates a serial cable to provide a simple substitute for existing old RS-232 serial lines. SPP defines how to set up virtual serial ports and connect two Bluetooth enabled devices. From the guest OS point of view, the endpoints of the SPP profile are seen as COM/RFCOMM ports.

Similarly to the USB redirection tests described before, the first local wireless redirection test we have carried out is purely functional. After the pairing phase, the remote Bluetooth USB dongle is perceived as local at the ISP side and seen as a RFCOMM port by the applications. The BT dongle inside the Smart Home receives periodical temperature measurements taken by a sensor that is connected to an Arduino board. These measurements are available at the ISP side thanks to the USB tunneling mechanism. In fact, in order to get the readings it is enough to display the arriving data using a common “cat” command as shown in Fig. 7.

In order to get a better understanding of what the user perception would be in a smart home scenario, we have also carried out a set of round trip delay measurements. Again, an Arduino board with a Bluetooth interface communicates with a control application through a USB dongle. We have tested three networking scenarios: local, same network and Madrid-Alcalá (see IV.A). In the first one, both the USB dongle and the application are located in the same machine. In the other two cases, the traffic of the USB dongle is virtualized and transported to the cloud by means of USB/IP tunnels.

Since what we want to assess are the effects of virtualizing the wireless home interface, the control application just responds back in a ping/pong communication schema. More precisely the procedure is as follows:

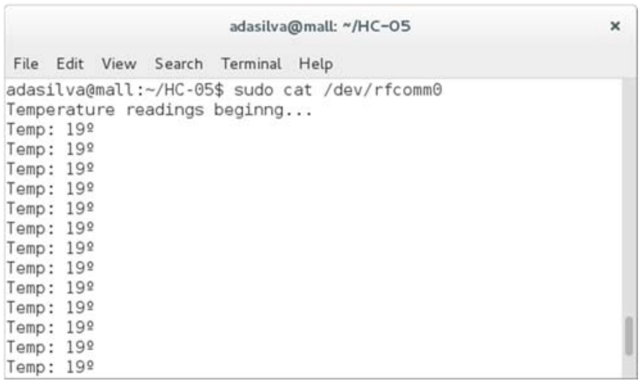
```
Round Trip Delay Measurement procedure

While( 1 hour )
        Arduino: set a digital pin up
        Send a ping signal through BT interface
        Wait for response pong
                Arduino: set the digital pin down
        End Wait
        Wait 1 second
End While
```

This way, we obtain a Pulse Width Modulation (PWM) logical signal on the Arduino digital pin that is set up and down by the algorithm above, where each high time corresponds to the round trip time (RTT) of one inquiry. We have arranged for this PWM signal to be read by an oscilloscope and the relevant data stored for further analysis.

Taking one measurement per second for an hour gives us 3,600 round trip times, from which approximate statistical behavior can be estimated. The results are depicted in Figure 8. Fig. 8 shows the raw results of the measurements, whereas Fig. 9 represents the estimated probability density function (PDF) of the round trip time observed in each networking scenario. In addition, Table II below summarizes the main statistical indicators of each RTT measurements series.

By comparing the RTT measurements of the "local" and "same network" scenarios, it is clear that there is no significant difference in their mean values and dispersion. The conclusion is that the delay overhead introduced by the USB/IP tunneling itself is negligible. The vast majority of these RTTs are between 15 and 50 ms.



```
adasilva@mall:~/HC-05$ sudo cat /dev/rfcomm0
Temperature readings beginng...
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
Temp: 19º
```

Figure 7. Common serial terminal running in Linux guest OS showing temperature readings.

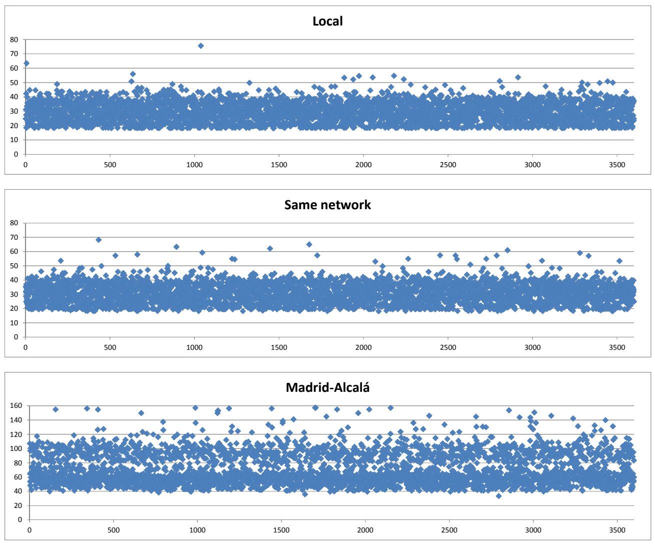


Figure 8. Round trip time measurements in ms.

In the Madrid-Alcalá scenario, the RTTs are considerably higher, ranging in their majority between 40 and 120 ms. Dispersion of these values is also bigger than in the other two cases.

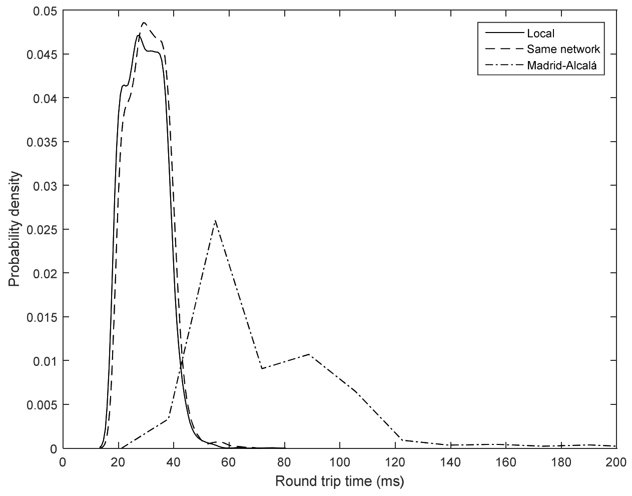


Figure 9. Estimation of the round trip time PDF of each scenario.

TABLE II. STATISTICAL INDICATORS OF RTT MEASUREMENTS (IN MILLISECONDS) FOR EACH LOCAL-REMOTE CONFIGURATION

	Local	Same network	Madrid-Alcalá
Max	75.6	68.2	1682.3
Min	18.1	18.1	33.5
Average	29.5	30.8	74.1
Std. deviation	6.9	6.9	43.5

Whether these RTT values are acceptable or not depends on the specific IoT application. For applications such as the usual non-critical in-home automation and control, these round trip delays are perfectly acceptable. Indeed, in [5] several applications of the "smart buildings and living" category are considered (among them, building condition monitoring, anomaly detection, lighting control, energy & water use and indoor climate control), and their tolerance to delays range from a few seconds up to a minute. In fact, each of the RTT tests we have performed simulates a common situation by which an event raised in the smart home environment (e.g., a temperature reading) is sent to the cloud, where it is processed and, as a result, an actuation order is generated in response and sent back home (e.g., an order to modify the settings of the HVAC system). This constitutes an example of what can be considered an indoor climate control service.

For more time-sensitive applications (such as alarms that have to be raised very shortly after the triggering event has occurred), it might be necessary to run part of the functionality inside the HRH as a lightweight container.

V. Conclusion and Future Work

This paper describes a basic implementation that shows that the HRH strategy is feasible through the tunneling of USB blocks. The proposed approach, based on common hardware and open software, has several benefits such as its low-cost, low-complexity, easy programmability and alignment with some of the current networking virtualization trends.

We have obtained both functional and non-functional indicators to assess the feasibility of our proposal. It is possible to virtualize and execute remotely basic applications that work as if the USB sources of information were connected locally to the same device. The tests we have made with a USB pen-drive, a conventional WebCam and a Bluetooth USB dongle prove this.

Moreover, USB/IP tunneling, even when combined with a local-remote scenario in which an actual residential network is involved, does not introduce a round trip delay that would impede in any way the deployment of conventional (non-critical) residential control and automation applications.

However, several important issues need further research in order to improve some aspects, for instance the management of uplink communications bandwidth between the HRH and the remote virtualization infrastructure, and the security considerations of a wide scale deployment.

As future work we aim to implement a complete prototype of the proposed architecture and its seamless integration in an existing SDN/NFV infrastructure including its standardized control plane protocols. This will help to assess, among other things, how the control plane affects the initial and subsequent delays observed by the applications, and whether the establishment of the USB/IP tunnels is feasible with the current SDN state of the art protocols. As regards the distribution of the virtualized functions, and given the specific QoS requirements and traffic characteristics of video traffic, we consider video preprocessing an ideal candidate to be runtime deployed in the HRH Docker container as a VNF. This way, if the overall composed video processing service can be divided into modules with clear interfaces, it could be easily deployed across the NFVi infrastructure, including the HRH. In our opinion, the demonstration of this scenario would constitute an interesting proof of concept mixing the service chaining capabilities often mentioned for composite virtualized services and the seamless distribution of components, that is one of the basis of fog computing.

Internet of Things has important security considerations, especially regarding data privacy. This is undoubtedly very important in the case of residential IoT, where concrete measurements are taken inside home, and it becomes even more critical if those data are sent to an external entity (the ISP), as happens in our architecture. Thus, confidentiality and authentication mechanisms have to be in place in any real deployment, and both vendors and NFV providers have to collaborate to create a secure and protected environment with privacy guarantees for users. This trustworthy ecosystem, necessary in our proposal, is not very different from what is expected of existing cloud-based IoT solutions, although it adds an additional stakeholder to the user and the IoT vendor: the VNF provider (or ISP).

On the other hand, the introduction of VNF and SDN brings both security challenges and enhancements [23]. On the positive side, time to upgrade the software with security patches, service availability, incident response time and real-time scaling are all factors that can be enhanced by executing the functionality at the ISP side, since the economies of scale allow for faster reaction to possible attacks and for more efficient and sophisticated preventive actions. In fact, the security functions themselves can be virtualized and executed in a manner that is independent of the specific hardware that enforces them. On the negative side, new threats linked to the virtualization architecture appear, such as hypervisor vulnerabilities (that could allow an attacker to modify some of the VNFs) and specific attacks to SDN controllers.

The performance of the proposed solution will also be subject to more study. Several use cases that include diverse types of services with different traffic requirements will be implemented and tested. Some parameters for the analysis will be: network latency, bandwidth and computing requirements.

Acknowledgment

This work has been supported by Spanish Ministerio de Economía y Competitividad under the grant TEC2015-67834-R.

References

[1] A. da Silva Fariña, A.B. García Hernando, and M.L. Mouronte López, "Residential Wireless Interfaces Virtualization: a Feasibility Study", The Thirteenth International Conference on Wireless and Mobile Communications (ICWMC 2017), Nice, France, July 23 - 27, 2017, pp. 67-72, ISBN: 978-1-61208-572-2.

[2] ETSI GS NFV 002 V1.2.1, "Network Functions Virtualisation (NFV); Architectural Framework", Dec. 2014.

[3] ETSI GS NFV-SWA 001 V1.1.1, “Network Functions Virtualisation (NFV); Virtual Network Functions Architecture”, Dec. 2014.

[4] Open Networking Foundation, “Software-Defined Networking (SDN) Definition”, 2018. [Online]. Available from: https://www.opennetworking.org/sdn-definition/ 2018.05.22.

[5] J. Mocnej, A. Pekar, W. Seah, and I. Zolotova, “Network Traffic Characteristics of the IoT Application Use Cases”, Technical Report ECSTR 18-01, Technical Report Series (ISSN 1179-4259), School of Engineering and Computer Science, Victoria University of Wellington, 6 January 2018. [Online]. Available: https://ecs.victoria.ac.nz/foswiki/pub/Main/TechnicalReportSeries/IoT_network_technologies_embfonts.pdf 2018.05.22.

[6] T. Zachariah et al., “The Internet of Things Has a Gateway Problem”, In Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15). ACM, New York, NY, USA, 2015, pp. 27-32.

[7] Etisalat, “Evolving the service provider architecture to unleash the potential of IoT”. [Online]. Available from: http://www.etisalat.com/en/system/docs/whitepapers/Etisalat-IoT.pdf

[8] Open Networking Foundation, “Specifications”. [Online]. Available from: https://www.opennetworking.org/software-defined-standards/specifications/ 2018.05.22.

[9] R. Morabito, “Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation”, IEEE Access, 5, 2017, pp. 8835 - 8850.

[10] T. Hirofuchi, E. Kawai, K. Fujikawa, and H. Sunahara, “USB/IP - a Peripheral Bus Extension for Device Sharing over IP Network”, 2005 USENIX Annual Technical Conference, pp. 47-60.

[11] ETSI GR NFV 001 V1.2.1, “Network Functions Virtualisation (NFV); Use Cases”, May 2017.

[12] N. Omnes, M. Bouillon, G. Fromentoux, and O. L. Grand, “A programmable and virtualized network & IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges”, Intelligence in Next Generation Networks (ICIN), 18th International Conference on, Paris, 2015, pp. 64-69.

[13] M. Niedermeier and H. De Meer, “Constructing Dependable Smart Grid Networks using Network Functions Virtualization”, Journal of Network and Systems Management 24, 2016, pp. 449-469.

[14] R. Yu, G. Xue, V.T. Kilari, V.T., and X. Zhang, “Network function virtualization in the multi-tenant cloud”, IEEE Netw., vol.29, no. 3, pp. 42-47, 2015.

[15] A. Berl, H. De Meer, H. Hlavacs, and T. Treutner, “Virtualization in energy-efficient future home environments”, IEEE Communications Magazine, 47, 12, 2009.

[16] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi, “Clouds of virtual machines in edge networks”, IEEE Communications Magazine, vol. 51, no. 7, pp. 63-70, 2013.

[17] R. Flores, D. Fernández, L. Bellido, N. Merayo, J. C. Aguado, and I. de Miguel, “NFV-based QoS provision for Software Defined Optical Access and residential networks”, IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), 2017.

[18] H. Moens and F. De Turck, “VNF-P: A model for efficient placementof virtualized network functions”, 10th International Conference on Network and Service Man-agement (CNSM), 2014, pp. 418-423.

[19] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, “The dynamic placement of virtual network functions”, NetworkOperations and Management Symposium (NOMS), 2014, pp 1-9.

[20] R. Mortier et al., "Control and understanding: Owning your home network", Proc. IEEE 4th Int. Conf. Commun. Syst. Netw. (COMSNETS), 2012, pp. 1-10.

[21] M. Dillon and T. Winters, "Virtualization of home network gateways", Computer, vol. 47, no. 11, pp. 62-65, 2014.

[22] J. Jo, S. Lee, and J. W. Kim, “Software-defined home networking devices for multi-home visual sharing”, IEEE Transactions on Consumer Electronics, vol. 60, no. 3, pp. 534-539, 2014.

[23] A. Dutta, “Security Challenges and Opportunities in SDN/NFV Networks”, 2016. [Online]. Available from: http://docplayer.net/65655705-Security-challenges-and-opportunities-in-sdn-nfv-networks.html 2018.05.22.

[24] M. Pattaranantakul, Y. Tseng, R. He, and A. Meddahi, “A First Step Towards Security Extension for NFV Orchestrator”, SDN-NFVSec '17 Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2017, pp. 25-30.

[25] IGR, “Moving to Cloud RAN and the Potential of Edge Data Centers”, 2016. [Online]. Available: www.commscope.com/Docs/iGR_cRAN_White_Paper.pdf 2018.05.22.

[26] O. Simeone, A. Maeder, M. Peng, O. Sahin, and W. Yu, “Cloud Radio Access Network: Virtualizing Wireless Access for Dense Heterogeneous Systems”, Journal of Communications and Networks, vol. 18, no. 2, pp. 135-149, 2016.

[27] F. Ramalho and A. Neto, “Virtualization at the network edge: A performance comparison”, IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Coimbra, 2016, pp. 1-6.

[28] A. Madhavapeddy and D. J. Scott, “Unikernels: Rise of the virtual library operating system”, Queue 11 (December 2013), no. 11, 30:30-30:44.

[29] B. B. Rad, H. J. Bhatti, and M. Ahmadi, “An Introduction to Docker and Analysis of its Performance”, International Journal of Computer Science and Network Security, vol. 17, no. 3, pp. 228-235, 2017.

[30] M. Chung, N. Quang-Hung, M. T. Nguyen, and N. Thoai, “Using Docker in High Performance Computing Applications”, in 2016 IEEE Sixth International Conference on Communications and Electronics, pp. 52-57, 2016.

[31] Y. Huang, A. Stavrou, A. K. Ghosh, and S. Jajodia, “Efficiently Tracking Application Interactions using Lightweight Virtualization”, VMSec '08 Proceedings of the 1st ACM workshop on Virtual machine security, 2008, pp. 19-28.

[32] L. Zeltser, “Security Risks and Benefits of Docker Applications Containers”, 2018. [Online]. Available: https://zeltser.com/security-risks-and-benefits-of-docker-application/ 2018.05.22.

[33] Docker Inc., “Docker Security”, 2018. [Online]. Available: https://docs.docker.com/engine/security/security/ 2018.05.22.

[34] CIS Docker 1.12.0 Benchmark, 2017. [Online] Available: https://www.cisecurity.org/cis-benchmarks/ 2018.05.22.

[35] The European Telecommunications Standards Institute, “Network Functions Virtualisation (NFV). Network Operator Perspectives on NFV priorities for 5G”, 2017. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper_5g.pdf 2018.05.22

[36] Y. Gittik “Distributed Network Functions Virtualization (White paper)”, March 2014.

[37] A. B. García, A. Da Silva, L. Bellido, F .J Ruiz, and D. Fernández, “Virtualization of Residential IoT Functionality

by Using NFV and SDN", In 2017 International Conference on Consumer Electronics (ICCE), Las Vegas, USA, January, pp. 8-10, 2017.

[38] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-Defined Fog Network Architecture for IoT", Journal Wireless Personal Communications: An International Journal archive, vol. 92, no. 1, 2017, pp. 181-196.

[39] N. Bizanis and F. A. Kuipers, "SDN and Virtualization Solutions for the Internet of Things: A Survey", 2016, 4, IEEE Access, pp.5591- 5606.

[40] S. M. M. Gilani, T. Hong, W. Jin, G. Zhao, H. M. Heang, and C. Chuan, "Mobility management in IEEE 802.11 WLAN using SDN/NFV technologies", EURASIP Journal on Wireless Communications and Networking, 2017.

[41] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Cloud of Things for Sensing as a Service: Sensing Resource Discovery and Virtualization", In 2015 IEEE Global Communications Conference (GLOBECOM). IEEE, pp. 1–7.

[42] R. Flores, D. Fernández, and L. Bellido, "A user-centric SDN management architecture for NFV-based residential networks", Computer Standards & Interfaces, Vol. 54, Part 4, November 2017, pp. 279-292.