Software-Defined LANs for Interconnected Smart Environment

Conference Paper · September 2015

DOI: 10.1109/ITC.2015.33

10 authors, including:

Mathieu Boussard
Alcatel Lucent
29 PUBLICATIONS **248** CITATIONS
SEE PROFILE

Laurent Ciavaglia
Alcatel Lucent
40 PUBLICATIONS **218** CITATIONS
SEE PROFILE

Michel le pallec
Alcatel Lucent
20 PUBLICATIONS **87** CITATIONS
SEE PROFILE

Nicolas Le Sauze
Nokia Bell Labs, France
33 PUBLICATIONS **417** CITATIONS
SEE PROFILE

Software-Defined LANs for Interconnected Smart Environments

Mathieu Boussard, Dinh Thai Bui, Laurent Ciavaglia, Richard Douville, Michel Le Pallec,
Nicolas Le Sauze, Ludovic Noirie, Serge Papillon, Pierre Peloso, Francesco Santoro
Alcatel-Lucent Bell Labs France
Centre de Villarceaux, Route de Villejust,
91620 Nozay, France
Email: first.last@alcatel-lucent.com

***Abstract*—In this paper, we propose a solution to delegate the control and the management of the network connecting the many devices of a smart environment to a software entity, while keeping end-users in control of what is happening in their networks. For this, we rely on the logical manipulation of all connected devices through device abstraction and network programmability. Applying Software Defined Networking (SDN) principles, we propose a software-based solution that we call Software-Defined LANs in order to interconnect devices of smart environments according to the services the users are requesting or expecting. We define the adequate virtualization framework based on Virtual Objects and Communities of Virtual Objects. Using these virtual entities, we apply the SDN architectural principles to define a generic architecture that can be applied to any smart environment. Then we describe a prototype implementing these concepts in the home networking context, through a scenario in which users of two different homes can easily interconnect two private but shareable DLNA devices in a dedicated video-delivery SD-LAN. Finally we provide a discussion of the benefits and challenges of our approach regarding the generalization of SDN principles, autonomic features, Internet of Things scalability, security and privacy aspects enabled by SD-LANs intrinsic properties.**

I. Introduction

With the development of the Internet of Things (IoT) and the emergence of smart environments (smart homes, smart buildings, smart cities), connected devices are experiencing an unprecedented growth in terms of number, functional diversity and heterogeneity. As a consequence, the management of the network connecting these devices becomes a burden. This is exacerbated in the context of home networks wherein end-users themselves have to face such a complex administration role without having the required skills or time. Therefore the promise of seamless and opportunistic interactions between these devices is not met, missing many valuable experiences.

Considering these issues, we envision a world where networked systems and connected devices are operated at our service, without requiring us to care about what their capabilities are, where they are or how to make them inter-work. In this new world, silos between devices and networks have vanished and there is no more barrier to the fulfillment of our expectations in terms of digital interactions and experiences, particularly when mixing devices and resources within and across multiple IoT-based smart environments.

In this article, we propose an approach to fulfill this vision, based on a software which controls and manages elements of a smart environment through their virtual representation, and enables the creation of Software-Defined LANs (SD-LANs) that interconnect and compose them, by using Software-Defined Network (SDN) architectural principles. Such software could be provided by an ISP, which could then leverage tight links to core network operations, or by an OTT player in pure overlay.

The paper is structured as follows. Section II presents the relevant IoT context and analyzes the relevant state of the art on smart environment controllers and SDN, and the applicability of the latter paradigm to the IoT and home networking context. Section III introduces our SD-LAN concept. The underlying virtualization framework is described in section IV, while Section V defines a general SDN-based architecture for smart environment controllers. Section VI describes our *Majord'Home* prototype implementation for the smart home context through the scenario of a multi-home video service SD-LAN establishment. Finally, section VII discusses SDN evolution, autonomic management, scalability, security and privacy aspects of our approach.

II. Context and State-of-the-Art

A. IoT Context

Driven by vastly varying requirements and device capabilities, the networking technologies in use in IoT solutions are fundamentally heterogeneous, spanning from wireless (ZigBee, WiFi, 3G/4G, WSN, etc) to wireline (Powerline, fiber, xDSL, etc) with very different communication characteristics.

In real world deployments, IoT networks are moreover often composed by the integration of different sub-networks with their respective legacies. It is commonly agreed that IoT solutions shall use IP as a converging network technology. Due to the aforementioned heterogeneity, it is often very difficult to provide fine grained control on networking and association of IoT devices in the current Internet architecture, which explains the predominance of network-agnostic client-server or specific Wireless Sensor Network architectures.

B. SDN and its applicability to IoT

SDN is not a completely new and revolutionary idea, but results from different contributions from the networking research area [1]. It relies on the advantageous separation between the control plane and the data plane, and open interfaces to more easily program network elements. Recent advances in the

OpenFlow[1] protocol and associated *controller* implementations have turned these research works into industrial reality.

The generally admitted SDN controller architecture identifies southbound and northbound interfaces (e.g. [2], [3], [4]). The main function of the southbound interface is the abstraction of the underlying network resources for easy manipulation by the controller. Typical interfaces are OpenFlow and ForCES[2]. Different functions (e.g. an orchestrator) may be implemented in addition to the network controller. Such sets of integrated functions (including the network controller) are generally called Network Operating Systems (NOS), in analogy with IT Operating Systems (for instance, see NOX [5]). Above such a NOS, the northbound interface allows applications to derive high level policies and to apply them to the network. Those high level policies could be handled by the NOS as different tasks. Examples of northbound interfaces are Frenetic [6], Procera [7], Netcore [8].

Recently, several research investigations have focused on SDN to cope with the aforementioned heterogeneity in IoT and the resulting variety of network requirements ([9]). However, the authors do not detail that much IoT use cases enabled by SDN. [10] on the contrary proposes a generic/generalized approach based on a layered control architecture. As in our work, this architecture aims at dealing with non only network elements (NEs) but also different types of IoT devices, and at hiding complex network configurations from the users. It relies on four layers: task description (user-friendly interface), service description (mapping of tasks into services), flow scheduling (handling of service flows) and low level communication (including the configuration of NEs). However, the context of managing vehicle fleets with its related energy issues leads to very different implementations and optimization, in a centralized manner. In our work, the prime focus is to build dynamic and isolated network of devices relying on multiple smart space controllers, each being responsible for local configuration tasks and communicating with other controllers for the establishment of end-to-end private networks.

C. SDN for home networks

Home networks have considerably evolved during the last decades, and the number of devices they connect is exploding: from an initial situation with a few PCs, a set-top-box and a residential Home Gateway, we have seen the emergence of many multimedia devices (Network Attached Storage (NAS), media servers, tablets, smartphones, connected TVs, etc.), and we are now observing the arrival of consumer IoT devices such as connected thermostats, weather stations, etc. Integrating these different devices into home networks is challenging as there are two contradicting requirements to increase their acceptance ratio: plug-and-play/easy to use operation vs. tight control of the information flows to enforce user privacy (e.g. controlling who accesses the devices and where collected information is sent). UPnP/DLNA is a perfect example of this dilemma for multimedia devices: all UPnP/DLNA devices in a home network can discover each other, so that if you give access to your home network to guests, they may have full access to you home video server from their smartphones!

In that regard, an interesting characteristic of SDN applied to the connected home is to greatly facilitate the home network management by virtualizing the home network infrastructure. [11] proposes to *slice* the common physical infrastructure between multiple service providers (an energy provider, an Internet Service Provider (ISP), an over-the-top video provider, etc.) and users (a guest network slice). Authors of [12] envision to control the usage of different persons at home based on user pre-defined cap usage rules.

Closer to our work, [13] also aims at building communities of users sharing home devices and services in their home networks. On the contrary to the aforementioned works, [13] does not explicitly refer to SDN, and relies on the exchange of home capabilities with peer home controllers thanks to an instant messaging and presence (IM&P) infrastructure. An Home Area Network (HAN) agent performs network access control in each HAN as low level configuration orders resulting policies defined in a Federal Relationship Manager (FRM). This FRM is central to enable communities of trusted users willing to share services and devices.

Finally, [14] presents an inter-home video service delivery through a SDN approach. A control box located in the cloud – administratively isolated from home networks – acts as a coordinator receiving user requests and translating them to application description messages sent back to home gateways. The latter host SDN home controllers and data-plane functions, and are interconnected using Layer 3 VPN and configured in order to support IP multicasting from one home to another. Accordingly the presented solution suffers from some drawbacks, especially related to Network Address Translation (NAT) traversal and device discovery in an inter-home context.

D. Middleware solutions for smart environments

While the above works focus on home networking, previous smart home and more generally smart environment research has investigated the concept of a general *home/environment controller*, as a software solution to organize and orchestrate connected resources and services of a given physical environment.

Among the precursors, the GAIA middleware [15] relied on a CORBA middleware implementation, so that active space system components could be implemented as distributed objects with CORBA IDL interfaces. More recently, HomeOS [16], proposed by Microsoft Research, relies on an Operating System paradigm to connect distributed smart environment devices through device drivers centralized at an orchestration point. The used abstraction is a collection of method signatures. Berkeley's Building Application Stack (BAS, [17]) and accompanying Building Operating System Services (BOSS, [18]) rely on RESTful APIs and a method based system abstraction to interact with heterogeneous devices. The Distributed Smart Space Orchestration System (DS2OS, [19]) at TU München uses hierarchically structured tuples that form a service model as abstraction in its Virtual State Layer middleware [20]. Communication here is not based on remote procedure calls but on distributed tuple space mechanisms.

Finally, tenants of the so-called *Web of Things* (WoT) have been relying on Web principles and technologies, to facilitate application development by leveraging largely used practices

[1]See Open Networking Foundation, https://www.opennetworking.org/.

[2]See *Forwarding and Control Element Separation* working group at IETF, http://datatracker.ietf.org/wg/forces/charter/.

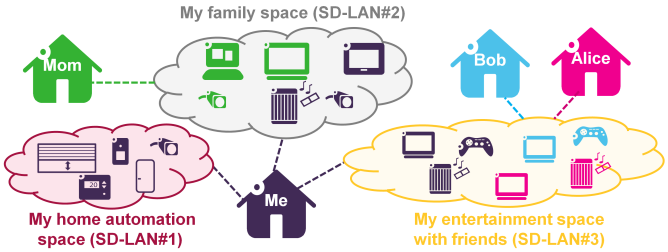

Figure 1. Example of possible SD-LANs within a home network and among multiple home networks.

and standards. Such approaches typically rely on Resource Oriented Architecture (ROA) through the use of RESTful design. This solves the problem of the need for standardization of interfaces (using HTTP verbs, user presentation through HTML-based representations, etc), while theoretically maximizing potential adoption by being accessible to the wealth of web developers [21]. We have previously proposed the WoT Framework [22] exposing both the environments and their constituents as Web resources, relying on the concept of Virtual Object (VO). However from a networking viewpoint, and similar to the later Internet of Things research, most of smart environment research have been focusing either on dedicated sensor networking solutions or as illustrated by the above examples on largely network-independent middleware solutions, focusing on programmability of the services offered by devices and the resulting ambient intelligence.

III. Software-Defined LANs

In order to fully develop smart environments, we believe that the control of devices and of the network has to be thought and realized jointly. This suggests the introduction of *Software-Defined LANs* (SD-LANs) that we define and discuss in this section.

A. Definition and characteristics

Software-Defined LANs (SD-LANs) refers to service-oriented and software-defined networks of devices. The term *LAN* (Local Area Network) here is slightly abusive as our SD-LANs aim at connecting devices across various administrative, applicative and technological domains. However, the term LAN also nicely designates a network area isolating inner devices from the outside, easing the discovery of new devices only in this limited area (e.g. UPnP/SSDP discovery). Therefore, *Local* refers here more to the controllable devices (and associated authorized users) within the SD-LAN than to a specific geographical aspect or set of used technologies. For instance, an end-user should be able to simply request a service, resulting in the creation of a SD-LAN connecting multiple objects in various locations (and therefore involving different networks), as if they were on the same LAN. This process should be as easy and user-friendly as possible, hiding unneeded complexity and configuration through automation.

SD-LANs are service-oriented, meaning they are typically inferred from a service request issued by a user or a software-agent (on behalf of a user, e.g. behaving according to pre-defined user policy rules). They represent virtual/personal networks including the connected devices the user wants to associate and possible additional services and functionality from ISPs (e.g. a security or QoS level) or third parties.

Figure 1 presents three examples of SD-LANs defined by a group of users, on top of their respective home network infrastructures: SD-LAN#1 for all home automation equipments of one home, SD-LAN#2 for shared storage and communications between two homes and SD-LAN#3 for entertainment with friends including devices from several homes.

These SD-LANs may require service-specific network configuration to be enforced among/between the different elements, but should not require any human to directly configure the network. Their spontaneous and end-to-end nature implies that the underlying networks that they span are configured as dynamically as possible to accommodate the necessary configuration, typically through SDN principles. This drastically contrasts with today's networked service deployment, which often requires long and tedious network provisioning.

SD-LANs are highly dynamic in nature: they can be created, reconfigured (e.g. adding/deleting components, changing configuration) or destroyed on the fly, upon user request or by the system upon condition changes (e.g. when the system detects that a device becomes unavailable). SD-LANs can have different lifespan depending on usages. For example, SD-LANs #1 and #2 shown on figure 1 are semi-permanent, while SD-LAN#3 is mounted and dismounted on-demand.

SD-LANs provide fine-grained control over which devices get to communicate and how, resulting in advanced security properties. For instance, SD-LANs intrinsically provide network isolation and minimize the surface of attack, especially compared to the state of the art of local networks (i.e., permanently open ports or channels to external cloud services).

SD-LANs are different from previously mentioned initiatives using SDN in home networks: we use SDN to define personal networks where users select the reachable devices and that the system self-configures to effectively allow such communications. Previous works on network slicing will however be relevant when analyzing isolation properties and competition among SD-LANs on a shared network infrastructure.

B. Application domains

The concept of SD-LAN allows supporting scenarios in any domain where devices need to communicate to deliver a service. SD-LANs enable the easy implementation of smart networks within and between various smart environments, e.g.:

- In the home domain, they allow to put users in control of their home network while removing the burden of network management tasks exacerbated by the proliferation of connected devices. They also unleash unprecedented or drastically simplified application scenarios such as making two devices between two homes communicate through a simple user request.
- In the enterprise domain, SD-LANs facilitate accommodating Bring Your Own Device (BYOD) behaviors, while enforcing network security by facilitating the creation by both end-users and administrators of specific isolated *slices* with well-defined characteristics. They can advantageously be used in smart building scenarios to isolate fleets of devices while enforcing specific networking characteristics among them.
- In the smart city domain, various silos can be enforced (e.g. for security/privacy purposes) or on the contrary

intersected with a fine-grained control (by instantiating SD-LANs encompassing selected devices from different silos, and only those).
- More interestingly, and when authorized, SD-LANs can be created across these various domains and actors, for instance to contribute a private home device to a fleet of smart city devices or to define a short-lived connection between a corporate device and a home device with a specified Quality of Service (QoS).

IV. Virtualization framework

In order to control and manage the devices and the network in smart environments, the relevant entities must be identified with their interactions, and abstracted in virtualized entities that will be managed. In this section, starting from the first definitions we gave in [23] within the context of home networks, we refine and generalize the concept to any smart environment.

A. Connected Objects, Virtual Objects and Avatars

The most generic view of smart environments is simply a logical network made of nodes interconnected by links, the nodes being objects that can produce, receive, forward or process data (source, sink and intermediate nodes). So we define the following concept of *connected object*:

Definition—Connected Object (CO): *A Connected Object (CO) is an entity that can generate, receive and/or impact data flows carried through the network of a smart environment.*

Thus a CO can be anything: a physical object, an application, a piece of software, a router, etc. In this paper, for switches or routers, i.e. for an intermediate node forwarding data and possibly processing it, we prefer to use the term of **Network Element (NE)** (as in SDN [4]), while keeping the CO term for other objects such as the *end devices* with which one can interact through the network.

We define the abstraction of a CO that will be used by the smart environment controller as follows:

Definition—Virtual Objects (VO): *A Virtual Object (VO) is an abstract representation of a CO that is used by the control and management software.*

The VO has all the properties required to enable the control and the management of the CO, such as its communication protocols, the CO availability, its capabilities, etc. It may also have methods that enable some specific processing, e.g. allowing the expected usage of the CO, or making the VO acting as a proxy for the CO when it has limited capabilities or is not available.

Each CO belongs to some user(s) and can be used by other users. To represent the rights of users on the different COs within their VO representations, we define *avatars* as follows:

Definition—Avatar: *An avatar is the representation of a user, mainly used to manage the rights of this user over its COs through the corresponding VOs within the control and management software.*

When a user manages his VOs through a user-friendly interface, the avatar can be simply the user's *login*. When required, the *avatar* may be more complex, being able to act on behalf of the user.

B. Communities of Connected and Virtual Objects

A SD-LAN as defined in section III interconnects some COs to render a service to a set of users, therefore one needs to consider groups of COs, and their virtualization. Unlike what we defined in [23], we prefer here to separate the notion of group of COs from its virtual representation, which leads to the two following definitions:

Definition—Community of Connected Objects (CoCO): *A Community of Connected Objects (CoCO) is a group of connected objects (COs) which have something in common, with something being anything.*

Definition—Community of Virtual Objects (CoVO): *A Community of Virtual Objects is a set of VOs which have been agreed to be interconnected for some specific purpose. So it is an abstract representation of a CoCO.*

The *anything* term in the CoCO definition can refer to a user's goal, in this case it corresponds to an SD-LAN as defined in section III. But it can also refer to other goals, such as administrative groups of COs to help the management of the COs, e.g. per user, per type, per environment, etc.

A CoVO inherits some of the properties and methods of the VOs that participate to it. It may have additional properties or methods specifically defined for the usage of the CoVO, e.g. specifying how the VOs interact together and in which order.

Note that avatars as defined above have an important role during the creation of a CoVO, where the inclusion of VOs requires some authorization.

V. SDN architecture for smart environments

The virtual entities defined in the virtualization framework described in the previous section can be used in the smart environment controller software. For each smart environment, we call *majordomo* this software, as it acts like a majordomo, managing the network and its devices on behalf of the owner(s) of the smart environment.

In this section, we define an architecture for such a *majordomo*, inspired from the architectural principles of Software-Defined Networks (SDN) as defined in [4].

A. SDN principles applied to smart environments

In our approach, the majordomo is a piece of software that may belong to a third party that we call the *operator* of this smart environment, while letting the environment owner keep a high-level control on what happens through user-friendly interfaces. This split of responsibilities is compatible with the use of a SDN-derived smart environment controller, where the data plane belongs to the place owner, while the control plane belongs to the operator of the smart environment. This requires however to update the SDN architecture of [4] as follows.

The traditional SDN **data plane** "comprises a set of one or more network elements, each of which contains a set of traffic forwarding or traffic processing resources". Compared to our context, this definition may be insufficient as, in addition to

classical switches or routers, we should consider end-devices without routing or switching capabilities, but only transmitting and/or receiving data. Therefore one can enlarge this definition of data plane to any CO as defined in section IV.

The **control plane** "comprises a set of SDN controllers, each of which has exclusive control over a set of resources exposed by one or more network elements in the data plane". These controllers are well adapted to NEs such as switches or routers. We need a similar functionality for COs that are end-devices. We differentiate here the ***Network Controller*** that is a classical SDN controller and the ***VO Controller*** which is an extension of the SDN controller to the abstractions of these COs, i.e. the VOs defined in section IV.

The VO controller may have some specific properties or actions due to the VO characteristics that must be handled and that are not considered within the classical SDN approaches, e.g. with OpenFlow. An example of such properties is the encoding format of the data that the corresponding CO transmits and/or receives, which must be set when the CO supports different possible formats.

The **application plane** "comprises one or more applications, each of which has exclusive control of a set of resources exposed by one or more SDN controllers". This should be extended by adding the VO controllers that we defined above.

Once we add the CO and the corresponding VO controller entities, all the other elements defined in [4] can apply to our context, and the SDN approach can be fully used.

In particular, the *recursive hierarchy* of [4] is useful to handle CoCOs through their virtualized abstractions CoVOs defined in section IV. A specific SDN controller dynamically handles all the CoVOs that are defined within the smart environment: the ***CoVO controller***. Following the SDN recursive hierarchy principle, its *data plane* is made of the network controllers and VO controllers that we defined above. The control commands follow the tree hierarchy, without synchronization issue: any entity (NE, CO, controller) is controlled by a single controller on top of it.

B. Control and management functions

In addition to extending the set of entities part of the data, control and application planes, the scope and functionality of the control and management planes should be augmented accordingly.

Therefore, in our context, we extend the definitions in [4] to end-devices as follows (additions in ***bold italic***):

1) Control "encompasses operations performed by a client or performed by a server on request by a client, for example operations between SDN controllers and NEs", ***COs*** "or applications";
2) Management functions are grouped under the OSS (Operation Support System, which is the term used in [4] for the block containing all the management functions) and "encompass operations to support the infrastructure, for example operations between SDN manager and SDN controller", ***CO*** "or NE".

Through the allocation of resources and policies to particular SDN clients or applications, and the provisioning of information necessary to allow separated functional entities to communicate, the OSS realizes the key interfacing and consistent interactions between the *end-user/owner* space and the infrastructure *operator* space. The classical service lifecycle functions such as equipment installation and maintenance, software upgrade, etc, of the OSS remain unchanged.

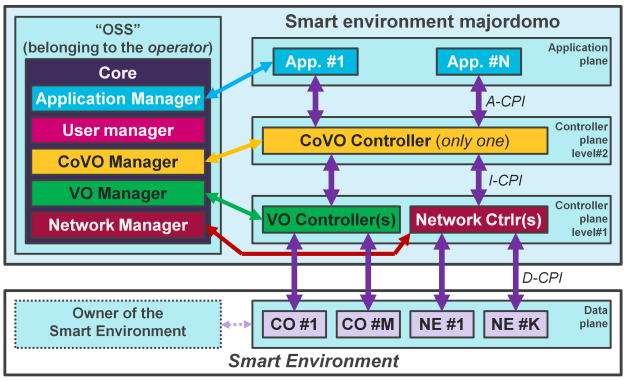

Figure 2. SDN-ized architecture of the smart environment *majordomo*.

C. SDN-ized majordomo architecture

Figure 2 represents the architecture we defined for the *majordomo* of a smart environment, following the SDN architectural principles previously discussed.

The data plane belongs to the owner of the smart environment. It is made of connected objects (COs), some of them being network elements (NEs) such as switches or routers.

The smart environment *majordomo* belongs to the operator of the the smart environment, who may or may not be different of the owner.

The *majordomo* includes two layers of SDN controllers. The first layer is composed of Network Controllers and VO Controllers, controlling respectively the NEs and COs through Data-Control Plane Interfaces (D-CPI, see [4]) by using their abstractions (VOs). The second level is made of *a priori* one single CoVO controller, through Intermediate-Control Plane Interfaces (I-CPI, see [4]), controlling the abstractions (CoVOs) of communities of connected objects (CoCOs).

Having a single CoVO controller per smart environment seems better than having several ones, because the *majordomo* should have a global view on the whole smart environment. For scalability issues (e.g. for large smart environments), if one chooses to split it into several controllers, then one should take care about the coordination between them.

The *majordomo* may include also some applications within an application plane. This concerns the applications that are downloaded within the *majordomo* by the operator, through the request of the user(s) for their usages. Applications supplied by external actors (e.g. within the cloud), are within an external application plane, not represented on figure 2. The application planes give their instructions and get information through Application-Control Plane Interfaces (A-CPI, see [4]).

Finally, the OSS includes all the management functions that are the counterparts of the controllers, i.e. Network, VO, CoVO and Application managers, as well as the specific part concerning the user management through their *avatars*. All these blocks are coordinated through the *core* part of the OSS.

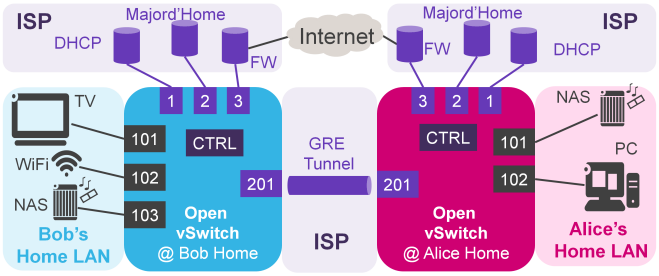

Figure 3. Residential gateways using Open vSwitches for our two-home demonstration scenario.

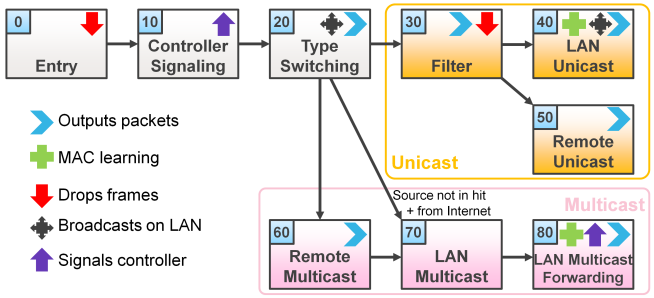

Figure 4. Flow tables for the OVS.

VI. SD-LANs for Home Networks

In this section, we apply the previously defined SD-LAN concept to the *smart home* context, describing the majordomo for home networks, nicknamed *Majord'Home*, that we developed as a first proof of concept. It is intended to demonstrate the SD-LAN concept and to serve as an experimentation platform for SD-LAN extension and generalization.

A. Proof of concept setup

Our experimental setup is composed of two home networks with their respective COs. Each one is connected to an ISP network emulated by three interconnected Alcatel-Lucent 7750 IP Service edge routers. Although a single ISP network is implemented here, we plan to capitalize on our previous work on assured-service quality connectivity tunnels between multiple carriers [24] to support multi ISPs scenarios.

Each home network is composed of a residential gateway (rGW) based on an Open vSwitch (OVS) to mainly ensure the forwarding function. This OpenFlow-configurable switch has been deployed together with a set of virtualized functions, namely DHCP, Firewall with NAT/PAT support between LAN and WAN, hence complementing the duties of the rGW (figure 3). To operate the rGW in an SD-LAN mode, we have additionally deployed the *Majord'Home* software solution.

B. The Majord'Home, a smart home controller

Following the virtualization framework of section IV, the *Majord'Home* first abstracts devices (COs) belonging to the various inhabitants of the home into VOs, in order to provide a programmable control of them through a common API. It also supports SD-LANs through the creation of service-oriented CoCOs and their virtual counterparts CoVOs. The *Majord'Home* follows the architecture defined in section V. Its implementation relies on a number of components as OSGi bundles, which, when started, expose a set of REST APIs (based on previous work described in [22]).

The VO manager maintains the list of VOs corresponding to the COs belonging to the local environment and their state. Although currently we use VOs as mere descriptions of the corresponding COs attributes and capabilities, they can also act as CO Controllers – in this case they may require a specific implementation under the form of CO driver bundle.

The CoVO controller interacts with the network controller to manage the SD-LANs according to the modifications of CoVO and/or CO states (Connect/Disconnect CoVO, CO presence). As a CoVO can span multiple locations and hence multiple *Majord'Homes*, it is particularly responsible for maintaining the state of the local part of a CoVO and notifying remote parts of this CoVO of changes.

The network controller configures the network functions (DHCP, NAT/PAT) and the network elements (i.e. the OVS embedded in the rGW) through an OpenFlow 1.3 interface. It is also capable of informing the VO manager of the availability of the COs by detecting their network presence. This information can impact connected CoVOs using this CO and trigger commands from the CoVO controller to the network controller to update existing configurations. In our smart home context, each SD-LAN has the same network properties emulating a standard Ethernet LAN: each CO in a SD-LAN can directly communicate only to other COs belonging to the same SD-LAN and broadcast traffic is authorized only within the SD-LAN boundaries[3]. At the initialization, the fail-mode of the OVS is set to *secure* thus blocking self-modification of flows when the connection to the network controller fails. Then 9 flow tables are created (see figure 4), each with a specific role:

1) Table 0 aims at filtering mis-formatted packets;
2) Table 10 informs the controller of newly discovered devices (includes MAC learning);
3) Table 20 differentiates multicast/broadcast from unicast traffic, and tags differently LAN-originated and WAN-originated traffic;
4) Table 30 tests pair of source and destination addresses to determine if a unicast frame is allowed or not;
5) Table 40 is updated through MAC learning and used to forward unicast frames in the LAN;
6) Table 50 sends unicast frames to the legitimate remote home networks;
7) Table 60 sends multicast frames to the legitimate remote home networks;
8) Table 70 sends multicast packets to the legitimate destinations in the LAN;
9) Table 80 is updated through MAC learning to forward multicast frames in new LAN multicast tree.

Conflicts on IPv4 addresses could easily be solved by double NAT, but some applications (e.g. DLNA) require specific proxies to overwrite private addresses present in the payload. SD-LAN properties thus naturally advocate for the use of IPv6.

The user manager deals with the user profiles and the user permissions used for access control (e.g. restricting the access to objects). Access control is enforced at each manager, i.e., whether the user is authorized to see, use or administrate the VO, CoVO or User manager (possibly at each element level).

[3] A CO may belong to several SD-LANs at the same time (if it makes sense for the object).

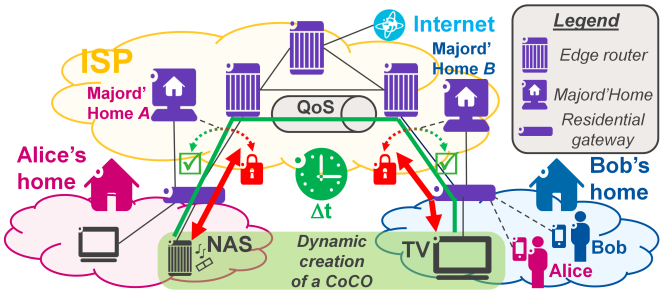

Figure 5. Overall demonstration scenario with the creation of a Community of Connected Objects between two remote homes.

The *Majord'Home* also implements eventing and notification mechanisms together with interfaces to communicate with ISP systems and with other remote *Majord'Homes*.

The end-user can interact with the *Majord'Home* through an Android client, which provides a simple user interface allowing the consultation of the different VOs belonging to known remote *Majord'Home* or to the local one, and enabling the activation of persisted CoVOs or the creation of new ones.

C. Scenario of demonstration

Figure 5 shows the scenario that illustrates the cooperation between two *Majord'Homes* (each one responsible for a distinct home) to create a temporary SD-LAN between two devices belonging to different home networks (a NAS in Alice's home including a DLNA server and a connected TV in Bob's home including a DLNA player) in response to a user request (e.g. Alice visiting Bob).

Upon Alice's arrival in his home, Bob provides her with the credentials to connect to his WiFi hotspot. However, Bob has previously configured (via his *Majord'Home*) a guest network policy consisting in providing to guests' devices a connectivity restricted to the Internet and to his own *Majord'Home*, disabling any other traffic. Alice's tablet can therefore access her *Majord'Home*, and to Bob's one. Naturally, through his *Majord'Home*, Bob has a different visibility on virtual objects in his home than Alice has. This is shown in figure 6 on Alice's tablet (bottom right) and Bob's smartphone (bottom left) views.

However, Alice can request use permission on elements (typically VOs) she could originally only see. The screenshot on the bottom right of figure 6 represents Alice's tablet screen after Bob has authorized her to use the Virtual Object related to his TV (whereas she can only see Bob's HomeNAS). Once Alice is able to use Bob's TV, she is able to create a CoVO including this TV and her home NAS, creating a logical association between the corresponding VOs.

When Alice launches a *Connect* command on the new CoVO, OpenFlow messages are triggered to set up the forwarding rules on each OVS, allowing the communication within the physical CoCO. In the case of the TV and NAS CoVO, both control (e.g. UPnP Simple Service Discovery Protocol [SSDP] messages) and data (e.g. video streams) traffic are enabled between the two COs. As a result, Bob's TV now detects Alice's NAS as a new DLNA device thanks to SSDP multicast over the newly set-up SD-LAN. Any device that is not part of a connected CoVO together with Bob's TV cannot be detected by the DLNA control point of the TV: in the demo, a dedicated CoVO gathering Bob's multimedia devices allows Bob's HomeNAS to be detected, while Alice's tablet, though being DLNA server capable, is invisible from Bob's TV as shown on figure 6. Note that there is no network transitivity property between the CoVOs: even if Bob's TV is included in a CoVO with Bob's NAS and in another one with Alice's NAS, both NASs cannot directly communicate.

As the physical CoCO spans across two different homes, the two *Majord'Homes* have exchanged dedicated information in order to achieve the setting of the corresponding multi-home CoVO. The information includes, for instance, the identification of the aforementioned VOs, the IP addresses used to establish the GRE tunnel described below and QoS parameters to be applied for this tunnel. Upon the *connect* command, the effective establishment of the GRE tunnel between the two OVSs is triggered if this tunnel was not preexisting for any other CoVO between these two homes. At the same time, the necessary OpenFlow messages are sent to each of the two OVSs by their *Majord'Home*, ensuring that the traffic transits between the homes through the GRE tunnel.

As a result of the SD-LAN, Alice can now use the TV to access her distant NAS content, as if the two devices were on the same LAN, selecting and playing movies with the native DLNA application of the TV. Upon leaving Bob's home, Alice can disconnect the SD-LAN, reverting the network configuration to its prior state.

This preliminary proof-of-concept therefore validates the concept of SD-LANs even between remote networks, providing both a dynamic programmability of the respective network elements and an isolation between subsets of the devices.

VII. Discussions

Before concluding, we discuss in this section some open questions and possible future works concerning some aspects of our SD-LAN approach about SDN evolutions, self-management, SD-LAN scalability, security and privacy.

A. Design choices for SDN-enabled smart environments

As explained in section V, SDN mainly addresses NEs such as routers and switches, using for example OpenFlow to control traffic forwarding, while additional features are needed to address any kind of COs. Indeed, when setting up a CoVO to interconnect two or more devices, what is important is not only the endpoints that send or receive data, but also if they can exchange such data: is the data format compatible (e.g., in the demo scenario of section VI, the connected TV and the media server are both DLNA/UPnP compatible)? Have they the capacity to do it in the current context (e.g., one device may be unavailable because it is already used in another SD-LAN)? Are they allowed to do it? These considerations lead to some questions that we left for further work:

1) Do we need to extend SDN by considering additional features for the control of end-devices? Or, alternatively, do we need to use the current version of SDN and make it interwork with other frameworks to control these end-devices?
2) What are the additional features to be attached to the VO description? (e.g., list of protocols, list of supported applications, current context of use, etc.)

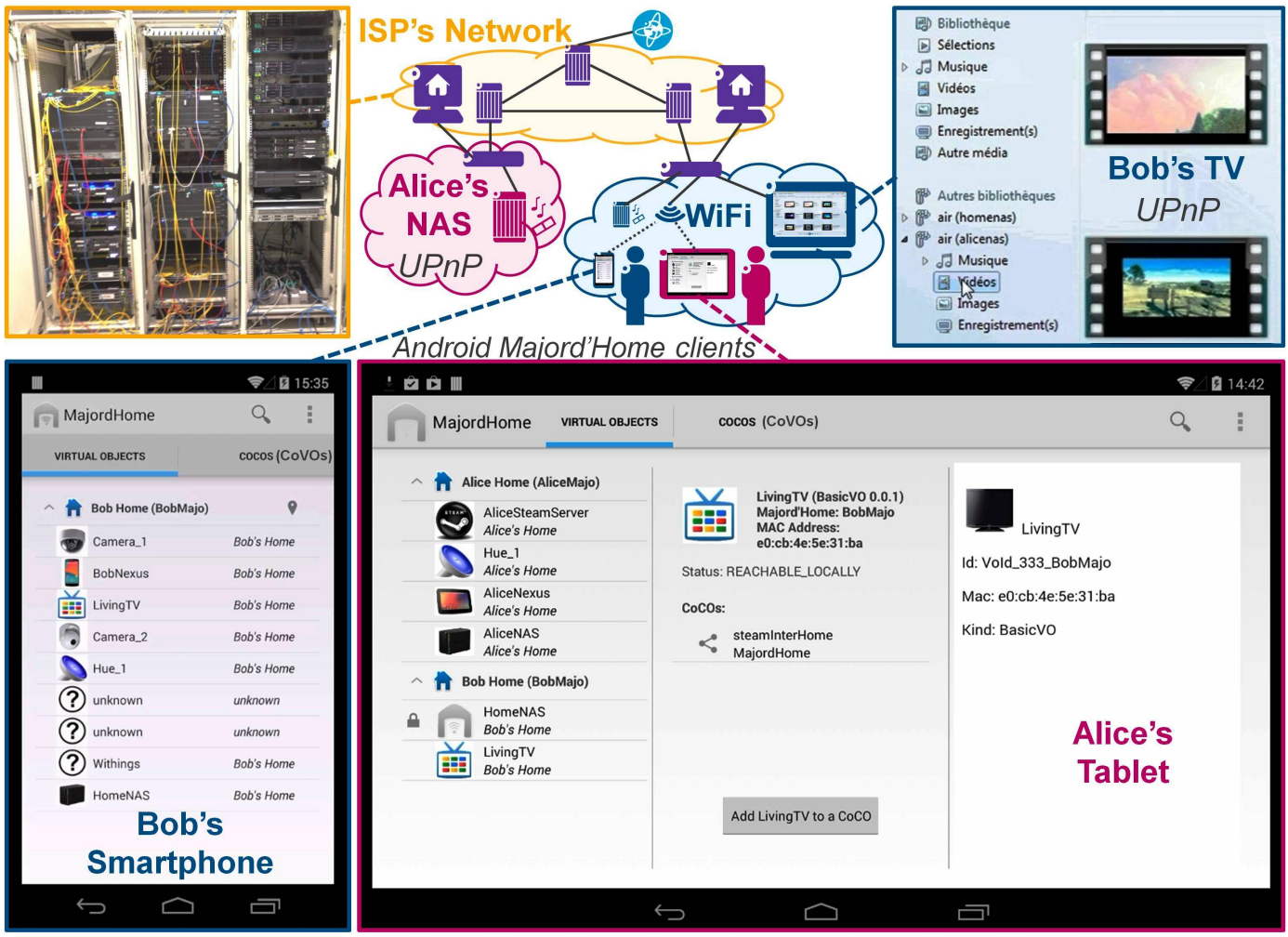

Figure 6. Elements of the *Majord'Home* demonstrator: routers and servers, TV screen snapshot, Android interfaces on smartphone and tablet.

B. Towards self-managing SD-LANs

As exposed earlier, the smart environments are intrinsically numerous, dynamic and heterogeneous. The management of such systems shall be capable of reactively and most important *proactively* adapt and scale their monitoring, diagnosis, discovery, coordination, fault-management functionality to match these characteristics and dynamics.

The evolution towards self-managing SD-LANs will enable both functional and performance gains through the introduction of principles such as automation, providing efficiency gain for repetitive and systematic tasks, autonomy to assess specific and different local actions, adaptation to cope with continuous context change, and actionable abstractions to enable coordination between heterogeneous entities.

C. SD-LAN scalability

SD-LANs may considerably transform the way network will be operated as we intend to have a fine grain control on communications between devices. In future works, the scalability of the concept will need to be addressed:

1) In each smart environment, the number of entries in flow tables is to be evaluated considering growing numbers of devices and service SD-LANs per physical spaces (and between physical spaces).
2) Individual flows will obviously be mapped into larger pipes within core networks following retail/wholesale services mapping as suggested in [23].

Tradeoffs to best place the various blocks of the *Majord'Homes* and distribute the flow of information and computing power have also to be evaluated.

D. Security and privacy considerations

SD-LANs provide users with fine-grained control over which COs get to communicate and how, resulting in interesting properties from a security and privacy viewpoint. Compared to the state of the art of local networks, SD-LANs provide better network isolation, as the easiness of defining SD-LANs allows for finer-grained, per service segmentation: a CO can not see any traffic from a SD-LAN it does not take part in. Through this isolation and dynamicity (e.g. a short-lived SD-LAN connected for the sole duration of the service, such as a video delivery), they also minimize the surface of attack. This allows in particular circumventing a number of security threats on poorly secured IoT devices as these are no more *easily* accessible to potential attackers.

Regarding further work on security, a number of items will be investigated. In our approach, SD-LANs can be multi-tenant: their COs can be provided by different users with different roles and can be dynamically added or removed. Thus the required access control models are very complex to design, while their practical implementation should be as user-friendly as possible to let the layman define them. Another challenge is the secure bootstrapping of the smart environment and possibly of selected SD-LANs and the definition of security handshakes across heterogeneous smart environment infrastructures in a way that ensures usability as well. A related challenge is the management of the secrets (especially their persistence) given the dynamicity of added or removed items in a SD-LAN.

From a privacy viewpoint, our approach enables users to explicitly tell the system upon SD-LAN creation which COs get to communicate (see demo scenario), and for example to define distinct SD-LANs for different levels of privacy. Furthermore, instrumentation of SD-LANs would allow for interesting information visualization challenges (to provide user interfaces allowing users to monitor exchanges), or even additional intelligent features (e.g. alerting users of abnormal traffic through pattern analysis and detection).

VIII. Conclusion

In this paper, we have presented a smart environment controller, which generalizes classical SDN controllers to the management of all types of smart environment resources. This enables the creation of *Software-Defined LANs* which are dynamic and service-oriented micro-networks of resources, blurring the physical and network boundaries of current environments. We have described the virtualization framework that allows to represent and manipulate these resources, and the possible mapping on the standard SDN architecture of the ONF. In order to prove the feasibility of the approach, we have implemented a smart home controller prototype and illustrated it in a multi-home DLNA video-delivery scenario. Finally, we discussed future directions for this work concerning the enhancement of this SDN architecture, autonomic management, scalability, security and privacy.

Acknowledgment

The authors thank their colleagues François Dorgeuille, Antony Martin, Rémi Varloot, Dominique Verchère and Martin Vigoureux for fruitful discussions about this work.

References

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[2] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[3] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[4] Open Networking Foundation (ONF), "SDN architecture, Issue 1," https://www.opennetworking.org/, Technical Paper, Jun. 2014.

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008.

[6] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," in *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming*, 2011, pp. 279–291.

[7] A. Voellmy, H. Kim, and N. Feamster, "Procera: A language for high-level reactive network control," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012, pp. 43–48.

[8] C. Monsanto, N. Foster, R. Harrison, and D. Walker, "A compiler and run-time system for network programming languages," in *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2012, pp. 217–230.

[9] A. L. Valdivieso Caraguay, A. B. Peral, L. I. Barona López, and L. J. García Villalba, "SDN: Evolution and opportunities in the development IoT applications," *International Journal of Distributed Sensor Networks*, vol. 2014, no. 735142, 2014.

[10] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *Network Operations and Management Symposium*, May 2014, pp. 1–9.

[11] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks*, 2011, pp. 1–6.

[12] H. Kim, S. Sundaresan, M. Chetty, N. Feamster, and W. K. Edwards, "Communicating with caps: Managing usage caps in home networks," in *Proceedings of the ACM SIGCOMM Conference*, 2011, pp. 470–471.

[13] R. Brennan, Z. Etizoni, K. Feeney, D. O'Sullivan, W. Fitzgerald, and S. Foley, "Consumer-managed federated homes," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 194–201, Jun. 2014.

[14] J. Jo, S. Lee, and J. Kim, "Software-defined home networking devices for multi-home visual sharing," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 3, pp. 534–539, Aug. 2014.

[15] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "A middleware infrastructure for active spaces," *IEEE Pervasive Computing*, vol. 1, no. 4, pp. 74–83, Oct. 2002.

[16] C. Dixon, R. Mahajan, S. Agarwal, A. J. Brush, B. Lee, S. Saroiu, and V. Bahl, "The home needs an operating system (and an app store)," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 18:1–18:6.

[17] A. Krioukov, G. Fierro, N. Kitaev, and D. Culler, "Building Application Stack (BAS)," in *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, 2012, pp. 72–79.

[18] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "BOSS: Building Operating System Services," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation*. Lombard, IL: USENIX, 2013, pp. 443–457.

[19] M.-O. Pahl and G. Carle, "Taking smart space users into the development loop: An architecture for community based software development for smart spaces," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, 2013, pp. 793–800.

[20] ——, "The missing layer – virtualizing smart spaces," in *IEEE International Conference on Pervasive Computing and Communications Workshops*, Mar. 2013, pp. 139–144.

[21] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of Things: resource-oriented architecture and best practices," in *Architecting the Internet of Things*, D. Uckelmann, M. Harrison, and F. Michahelles, Eds. Springer Berlin Heidelberg, 2011, pp. 97–129.

[22] M. Boussard, B. Christophe, O. Le Berre, and V. Toubiana, "Providing user support in web-of-things enabled smart spaces," in *Proceedings of the 2nd International Workshop on Web of Things*, 2011, pp. 11:1–11:6.

[23] M. Boussard, B. D. Thai, R. Douville, N. Le Sauze, L. Noirie, P. Peloso, R. Varloot, and M. Vigoureux, "The Majord'Home: a SDN approach to let ISPs manage and extend their customers' home networks," in *1st International Workshop on Management of SDN and NFV Systems, co-located with CNSM*, Rio de Janeiro, Brazil, Nov. 2014.

[24] FP7 ETICS consortium, "Economics and Technologies for Intercarrier Services," https://www.ict-etics.eu/, Final white paper, 2013.