An Open-Source Virtual Set-Top-Box for Softwarized Networks

Gabriele Baldoni, Alfio Lombardo, Sergio Micalizzi, Corrado Rametta, and Alessandro Vassallo
DIEEI – University of Catania – Catania, Italy (mail: name.surname@dieei.unict.it)

Abstract— *Network "softwarization" is gaining increasing popularity to achieve dynamicity and flexibility. Cloud computing, together with the new paradigms of SDN and NFV are supporting this evolution. However, the need to move services closer to users to guarantee low latency in the service fruition on one hand, and the trend to support personalization of services on the other hand, are stimulating a migration of the services toward edge nodes. This is the target of the platform proposed in the INPUT project to support Future Internet personal cloud services in a more scalable and sustainable way. The INPUT platform enables next-generation cloud applications to go beyond classical service models replacing physical smart home devices, with virtual entities, providing them to users "as a Service." In this paper, we discuss the design and open-source implementation of a virtual Set-Top-Box that is compliant to the INPUT platform.*

Index Terms— **SDN, NFV, Network Softwarization, Personal cloud services, Home Entertainment, Open Source.**

I. INTRODUCTION

The tremendous evolution of distributed software applications is determining a global rethinking of protocols and architectures in the current Internet. Network functions, ranging from performance-improving applications (e.g. WAN optimizers, proxies, and application gateways) to security appliances (e.g. firewalls, IDS), are become more numerous than routers and switches [1-2]. For this reason, Telco Operators are migrating their infrastructure from an implementation of network functions on hardware middleboxes, to very efficient software solutions based on their virtualization on general-purpose servers, following an approach used for the first time with open-source software routers [3-5]. This trend, also referred to as "network softwarization" [6], is supported by the two network paradigms of Software Defined Networks (SDN) [7-8] and Network Functions Virtualization (NFV) [9-10].

At the same time, the diffusion of "smart" devices, like smartphones and tablets, is raising some new, special requirements. In fact, their limited storage and computational capabilities have led to the development of a significant number of services on the cloud. However, in order to satisfy the low latency levels and bandwidth requirements posed by next generation cloud services, like online gaming [11], interactive e-learning [12], video surveillance [13-14], etc., the next frontier to reduce the end-to-end network latency will clearly consist of hosting cloud applications directly into network operators' infrastructures, and of being as close to end-users as possible.

Starting from the observation that current business models applied by Telco Operators today to provide home services imply deployment of new devices within the customer premises, including a residential gateway for Internet for Internet access and VOIP services, and a Set-top Box (STB) for multimedia services, the H2020 INPUT project [15] has proposed a platform that leverages on both softwarized network technologies and the edge-computing approach [16] to go beyond classical cloud-computing service models for home ICT services. Specifically, the INPUT project introduces the new concept of *personal network*, and allows the design and the deployment of virtual smart devices, i.e. virtual images that are able to replace physical Smart Devices (SDs) usually located in users' homes (e.g., set-top-boxes, etc.), providing them "as a Service" (SDaaS). The proposed approach overcomes the typical delay problems related to the use of either the peer-to-peer (P2P) approach [17-19] or remote data servers, as employed by standard over-the-top providers, by moving cloud computing and network services closer to the edge network nodes. Moreover, it aims at reduces energy consumption, that is one of the most big problems in the future of telecommunications networks [20-21].

The potential innovation of INPUT approach with respect to other previous research initiatives is related to the new dimension of personalization in the services and virtualization of network entities by using open source configurable devices as edge routers, which can be managed according to an in-network programmability approach. Moreover, thanks to the application of NFV at the edge of the network, it is possible to move the complexity of processing and data collection closer to the user, realizing a user-centric personalization of the Quality of Service (QoS) requirements.

Target of this paper is the description of a relevant INPUT use case, namely the virtual Set-Top Box (vSTB), which embraces the ETSI's Use Case 7 on the "Virtualisation of the Home Environment" [22], by focusing on virtualization of home DLNA-compliant appliances [23].

The paper is structured as follows. Section II will describe the reference architecture of the INPUT platform. The vSTB application is presented in Section III. Section IV will describe the open-source implementation of the vSTB, while Section V draws some conclusions and presents some future work.

II. NETWORK REFERENCE ARCHITECTURE

The infrastructure of a legacy telecommunication network with full technological convergence of mobile and wireline accesses is mainly composed of the Access Network, the Edge Network and the Core Network. Today, specific nodes in the Edge Network are the only devices that are in charge of

authenticating the customers, authorizing their traffic and services, as well as extracting information for pricing and billing purposes. In other words, they constitute the only network nodes where network services follow a "personal" paradigm.

The INPUT platform represents a novel Edge-Network infrastructure concept that integrates the SDN/NFV networking paradigm with a fog-computing framework together with their management functionalities, in order to provide *personal cloud services*.

Two important concepts in the INPUT platform are the virtual Smart Device and the Personal Network.

The *virtual Smart Device* (*vSD*) is a virtualization entity representing a physical Smart Device (SD), like for example a storage server, a set-top box, a video recorder, a home-automation control unit, a game console. A *vSD* can be used as a SD as a Service (SDaaS), and is able to replace the relevant physical device in the users' home networks.

A *Personal Network* is a secure and trusted virtual overlay network capable of interconnecting user's SDs (either virtual or physical) with standard layer-2 (L2) protocols and operations equivalent to the ones presently available in home network of the users [15]. In other words, a Personal Network is an extension of the user's physical home network, also including vSDs running in the Telco Operator network, with the possibility of "following" the user whatever his/her geographical location (inside/outside the user's home).

An example of deployment of a Personal Network is shown in Fig. 1. It is realized by virtualizing both the user's physical home gateway, by replacing it with a virtual home gateway (*vHGW*), and user's physical SDs with *vSDs*. Both *vHGW* and *vSDs* are launched as software instances in commodity computing facilities deployed in the INPUT platform running at the edge of the Telco Operator network, an infrastructure compliant with the SDN/NFV and the fog-computing paradigms.

The vHGW is realized by transferring the network functions that are typically provided by the user's home gateway (e.g., IP forwarding/routing, firewall, DPI, NAT, DHCP), into software instances, called *Net_Functions*.

A vSD, on the other hand, is obtained as a personal cloud service realized as a chain of one or more User applications (*User_App*), Service applications (*Service_App*) and Data Center applications (*DC_App*), each performing a specific task (e.g., user interface, web server, proxy, content caching, storage, computing, encryption/decoding, etc.).

A *Service_App* is a software instance running in a single "execution container" (e.g., a virtual machine) at the Edge Network, and provides application level services. *User_Apps* run on physical user clients (e.g., smartphones and tablets), and provide users with an interface to access personal cloud services. Finally, *DC_Apps* running on a remote datacenter, aim at completing the service with high-computational and storage-consuming facilities. This modularity allows personal cloud service providers to easily upgrade/extend the functionality of the vSDs.

By leveraging on data center and cloud technologies,

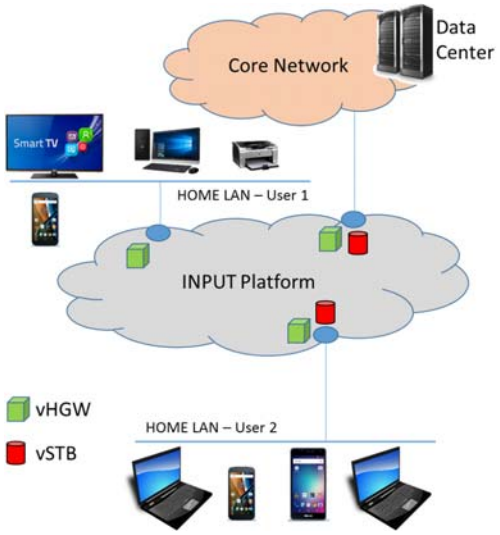

Figure 1: Mapping of the User's Personal Network into the INPUT Infrastructure.

Net_Functions and *Service_Apps* can be dynamically migrated from one computing facility to another one with the purpose of server consolidation and end-to-end latency minimization, thus reducing energy consumption and improving Quality of Experience (QoE).

The communication and information exchanged among different *Service_Apps* of the same personal cloud service are handled through a Back-End Network, an overlay network realized with a set of virtual L2/L3 interconnections built between two or more virtual/physical hosts, whose traffic is isolated from the one carried by the Personal Network or other Back-End Networks.

III. Virtual Set-Top Box Design

In this section we show how a virtual Set-top-Box can be designed over the INPUT platform to provide customers with a personal cloud service regarding multimedia entertainment at home. It consists of the virtualization of a set-top box (STB) device like the physical one provided today by content providers (CP) that offer services like pay-per-view of live video streaming and video on demand. Thanks to the SDaaS facility given by the INPUT platform, the STB is not physically present at the user home, but provided as a virtual STB (vSTB) through the INPUT infrastructure.

The vSTB considered in this use case provides the following main capabilities:

1. real-time streaming of multimedia content transmitted from a CP to one or more user's home player devices, such as smart TVs, and to one or more user's Smart Clients, such as smartphones and tablets;
2. recording of multimedia contents;
3. playout of multimedia contents previously recorded.

The potentiality of the STB virtualization becomes evident when the user leaves his own home network. In fact, in this case, the user can still exploit the vSTB personal cloud service using his Smart Clients to watch real-time and stored contents when he is in mobility.

Fig. 2 shows a high-level overview of the user's Personal Network when the vSTB service is active. We can distinguish between physical devices at the user side, i.e. in the user's *residential LAN domain*, and virtual devices in the *INPUT domain*, i.e. deployed inside the INPUT platform but belonging to the user's private domain. In addition to the above components, there are also other elements that can be considered in the *shared domain* because shared among a set of users, i.e. not belonging to any specific Personal Network.

The INPUT platform smart elements composing the service chains to realize a vSTB can be classified per application category as follows:

User_Apps

- *vSTB Configurator*: it is the mobile *User_App* enabling the user to access the configuration menus of the vSTB, interacting with the vSTB Manager *Service_App* described below;
- *Remote Controller*: it is a mobile application that the user installs on his Smart Client to remotely control the vSTB. It is connected to the VDI *Service_App* to access the CP content schedule, select a real-time or recorded content, and create virtual connections between the vSTB and the DLNA-compliant physical players that are available in the home LAN;
- *Event Notification*: it allows setting up a notification service to alert users when the streaming of a requested content is starting;
- *Mobile Player*: it allows watching video streams on a smartphone or tablet; it connects directly to the Personal Acquirer (PA) and the Digital Media Server (DMS) *Service_Apps*, both described below, in order to choose a real-time or a stored content.

Service_Apps

- *vSTB Core*: it is the core element of the vSTB, and represents the virtualization of a common physical STB device; it consists of four elementary Service_Apps, i.e. VDI, DMC, DMS, and PA, whose functions will be detailed in the sequel;
- *Edge Storage*: belonging to the shared domain, it is in charge of saving the recently recorded contents;
- *Edge Acquirer*: belonging to the shared domain, it is the *Service_App* receiving data flows from the CP through the dedicated Back-End network. It communicates with the PAs of all the users registered to the INPUT platform node where it is running, in order to publish the content timetable and transmit them the requested multimedia data flows. Moreover, according to messages received by the vSTB, it can perform flow rate control at both network [24] and application layer [25-27].

DC_Apps

- *vSTB Store:* it is a cloud service enabling the subscription of services/channels/events proposed by the CP. A user connects to this service every time he wants to change the terms of his subscription.
- *Historical Recording Storage:* contents saved by users are stored in a storage service within the INPUT nodes. When contents become obsolete and, thus, requested less frequently, they are sent to this *DC_App* to be stored in its storage system.

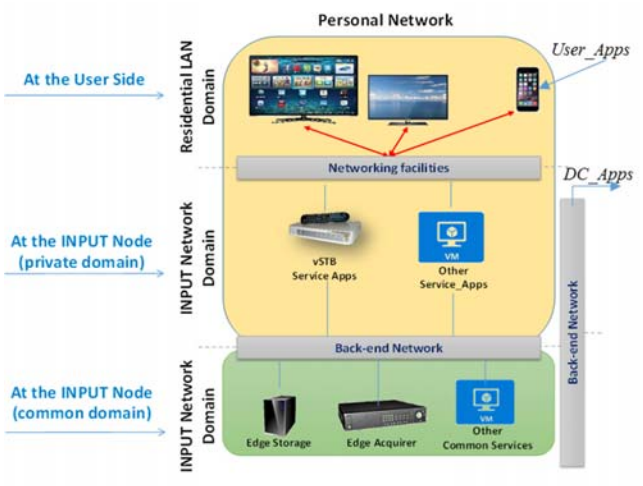

Figure 2: Personal Network when the vSTB service is active.

The vSTB Core is compliant with the DLNA standard, that is, it is able to communicate with other DLNA-compliant devices like smart TVs connected to the same personal network. It consists of four Service_Apps:

- *Virtual Decoder Interface* (*VDI*): this component represents the interface between the Remote Controller *User_App* and the functional components of the virtual decoder. By using the Mobile App on his own Smart Client (Android, iOS, Windows Phone or other frameworks), the user can select an action among view live contents, schedule a content recording and watch a recorded content from his own digital library. Furthermore, the VDI communicates with the Digital Media Controller to select the digital media player that the user wants to use to watch the video flows, and with the Personal Acquirer in order to select one of the available video streaming events to be played out or enable the recording of a selected event.
- *Digital Media Controller* (*DMC*): according to the DLNA standard, the DMC maintains the list of the DLNA players in the personal network, and communicates with the DMS to know the list of the stored contents.
- *Digital Media Server* (*DMS*): this element is DLNA-compliant too; it maintains and updates the list of contents exposed by both the PA and the personal digital library in the Edge Storage. Moreover, according to the encoding requirements received by the vSTB Manager, it performs transcoding to adapt the multimedia flows to the players. Let us stress that this peculiarity makes the difference between the INPUT and the legacy approaches. In fact, this allows the Telco Operator to customize the quality for each user thanks to the adoption of the fog-computing paradigm, and the over-the-top approach, which puts computation on remote clouds, therefore working on classes of users.
- *Personal Acquirer* (*PA*): it receives commands from the VDI to enable the live streaming of a content or the recording of a selected content. It communicates with the

Edge Acquirer at the common domain, requiring the multimedia flow selected by the user, or indicating that a specified event has to be forwarded to the Edge Storage to be recorded.

IV. Implementation

In this section we describe the implementation of some of the virtual functions described above, focusing in detail on the core elements composing the virtual smart device (see Fig. 3). All the produced libraries are available on line, as referenced in the following subsections.

A. *Implementation of the Virtual Decoder Interface*

The Virtual Decoder Interface represents a front-end server that is in charge of interfacing with the User mobile app, where the User_Apps are run. It is an open-source software realized in Java language handling all the RESTful requests coming from the mobile User mobile app. The latter connects to it in order to perform the following actions:

- Require the list of CPs;
- Require the list of available channels for a selected CP;
- Require a video channel to be watched by the Mobile Player in User mobile app;
- Require a video channel to be watched by a DLNA Player;
- Manage the playout of a recorded video stream (start/ stop/pause/resume).

The VDI also interacts with both the DMC and the PA as it will be better explained in the next sections. The VDI open-source library is available in [28].

B. *Digital Media Controller*

The Digital Media Controller acts as a controller of UPnP devices. It is in charge of discovering the DLNA players connected to the network and keeping the list of Digital Media Servers and the related contents updated. There are several applications available for both mobile and desktop environments, mainly developed in C++ or Java. These apps usually are based on two API open source libraries, "Platinum UPnP" [29] e "Cling" [30]. Our implementation of DMC [31] has been realized in Java by exploiting the Cling library, which is particularly suitable for the development of desktop and Android applications. It consists of two modules: Cling Core, to discover DLNA devices and the exposed services/features; Cling Support, to manage the data stream visualization on players.

The DMC receives the following requests from the VDI:

- List of DLNA Players;
- List of Digital Media Servers;
- List of Contents of a DMS;
- Play/Stop/Pause Content X on Player Y.

C. *Digital Media Server*

A customized Digital Media Server has been realized by using uShare [32], an open-source server software with the aim of being fully compliant with DLNA/UPnP clients. The uShare daemon serves media files (music, pictures, and video) to clients on a network. Example of clients include applications such as Totem and Kodi, and devices such as portable media players, smartphones, smart TVs, and gaming systems (such as PS4 and Xbox One). The DMS has been developed in order to receive live streams directly from the Personal Acquirer, and exposing them as stored files in the Personal Network. In this way, all the DLNA players connected to the Personal Network will be able to view both recorded files and live channels in the same way, and select them from a list of DLNA-server available contents.

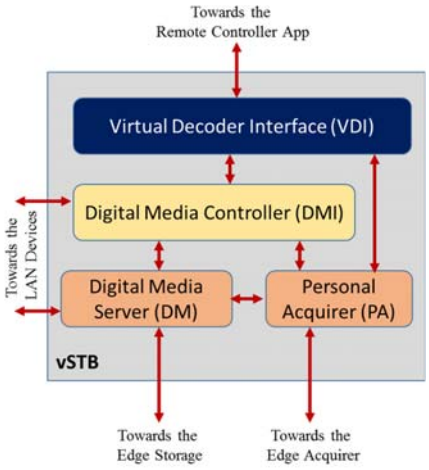

Figure 3. The core functions composing the virtual set-top box.

D. *Personal Acquirer*

The Personal Acquirer has been developed in C language [33], and is connected via two TCP signaling channels to both the Virtual Decoder Interface and the Edge Acquirer. The communication between VDI and PA is realized according to a client/server paradigm where the PA acts as server for the requests coming from the VDI. More in detail, the VDI sends the following request messages:

- Get list of available CPs;
- Get list of available channels;
- Get live stream (ID_Channel, ID_Output);
- Stop stream.

The communication between PA and EA follows a client/server scheme as well, where the EA acts as server for the requests coming from the PA. More in detail, the latter sends the following requests to the EA:

- Authenticate virtual smart device;
- Get list of subscribed CPs;
- Get list of subscribed channels;
- Get stream (ID_Channel);
- Stop stream.

On the base of the output selected by the User, the PA: 1) forwards the data stream to the Mobile Player; 2) creates a named pipe accessible to the DMS that, in such a way, can expose the selected live channel directly to the DLNA players and to the DMC.

E. *Edge Acquirer*

The Edge Acquirer is based on the open source software Multicat [34], a multicast and transport stream manipulation package timely modified in order to receive video data streams from the content providers and forward them to the personal acquirers. Our application, available at [35], as compared to the original library, listens on a TCP port data flow requests coming from the PAs and, according to the received requests, duplicates

incoming packets at run-time. In our prototype, the EA receives RTP/MPEG transport streams (each channel of each CP has a specific ID) and forwards them to the PA Service_Apps of all the connected vSTBs, according to the received requests.

F. User Mobile App

We developed the User mobile app as a mobile Android client app with the purpose of testing the platform functionalities. The mobile app has been developed in Android Studio v2.3 [36], ensuring frontward compatibility with Android 5.0 (Lollipop). It allows Users to play live video contents transmitted by CPs. Video transmission from CPs is based on the RTP protocol.
Users access content scheduling of each CP, so that they could choose to enjoy the streaming of an event. If an event is not still started, the user can subscribe the intent to watch it. The app will send a notification to the User a few minutes before the content streaming event starting time as a reminder. Furthermore, through this mobile app, users are able to request recording of a live event even if it is already started.
This mobile app is composed by four sections:

- Discover and show the CP list and, for each CP, the live video channels scheduled by it;
- Player, i.e. the section where it is possible to watch the requested live channel. As mobile media player, we have chosen Vitamio [37], an open multimedia framework for iOS and Android that guarantees compatibility with the most popular video codecs and communication protocols (in our prototype RTP has been employed as transport protocol).
- Remote Controller, a section enabling the view of a live channel directly on a DLNA-compliant player located in the User's Personal Network.

V. Testbed

Finally, in order to evaluate the effectiveness of the proposed implementation, we realized a proof of concept (POC) of it by using an emulation framework.

The latter has been realized by means of the interconnection of the *network emulator Mininet* [38] with real devices, such as wireless access points, 4G femtocell and smartphones. Mininet is an open-source network emulator that allows users to create virtual software-defined networks consisting of an OpenFlow [39] controller, flat Ethernet networks of multiple OpenFlow-enabled Ethernet switches, and multiple hosts connected to those switches.

The peculiarity of our platform consists in the capability of binding, by means of Python-language configuration scripts, OpenFlow switch ports to network interfaces of physical devices. In this way, they result directly connected to the emulation framework. This solution allows the interconnection of physical wireless Access Points (APs) and 3G/4G Radio Access Networks (RANs) to the emulated network and, consequently, any physical device (such as smartphone, pc or any other internetworking appliances) connected to the APs/RANs results as being part of the emulated network.

The network emulator also provides the platform with virtual hosts where, according to the VNF paradigm, application or network functions composing a Personal Cloud Service can be instantiated and run.

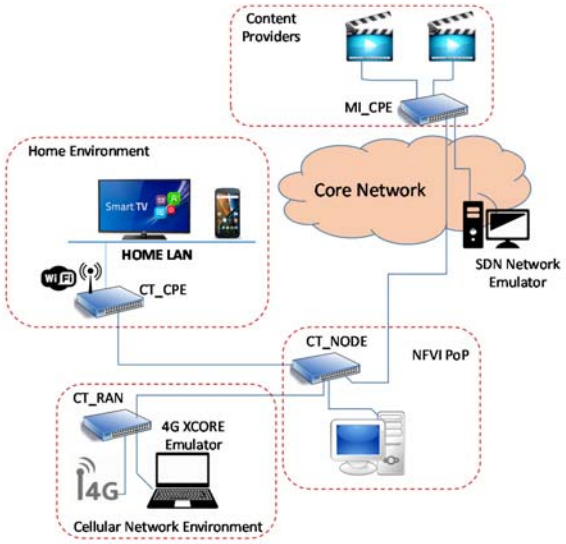

Figure 4. Testbed topology.

The *SDN controller* used in the proposed network topology is a customized version of *OpenDaylight* [40], a Network Operating System for SDN-NFV networks, developed under the auspices of the Linux Foundation and written in Java. It supports a variety of SDN protocols, so that it can be used for managing heterogeneous L2 switches. In the current demonstrator, it was used the version Hydrogen of OpenDayLight, which support OpenFlow 1.0 specifications to guarantee the compatibility with the Open vSwitches emulated by Mininet.

Finally, in order to provide the emulated platform with a mobile access network, a *4G femtocell in a box* has been connected to the platform. The attached femtocell, providing LTE connection to mobile devices moving in the area covered by it, was connected to the EPC Emulator Accuver XCORE [41] running in a Laptop and implementing the functions of the whole EPC infrastructure on a single PC, allowing in this way the possibility to test and develop solutions for mobile systems without the need of a real network infrastructure.

The testbed topology, shown in Fig. 4, consists of two Customer Premises Equipment, named CT_CPE and MI_CPE, one Radio Access Network node, named CT_RAN, four Network Functions Virtualization Infrastructure Points of Presence (NFVI-POP). Virtual functions composing the vSTB have been hosted in the NFVI-PoP closest to the emulated User, i.e. in the CT_Node. Two CPs have been emulated, each one transmitting three channels, and they have been connected to the node MI_CPE. User's home LAN (consisting of a Smart TV and a WiFi smartphone) has been connected to the node CT_CPE meanwhile the user's mobility has been emulated by connecting his mobile phone through the 4G RAN.

The proposed implementation of virtual set-top box has been successfully tested by means of this environment showing promising results compared with well know and widespread data center based approaches.

VI. Conclusions

In this paper we presented the design and implementation of a softwarized smart device, a virtual set-top box, compliant with the network "softwarisation" approach proposed in EU INPUT

project. The INPUT paradigm and the relative architecture have been briefly introduced in order to clarify the technical background and the network architecture behind the idea of enabling next-generation cloud applications able to replace physical Smart Devices, usually placed in users' homes, with virtual entities, providing them to users "as a Service."

The design and implementation of a Virtual Set-Top-Box, here described, is one of the use cases proposed inside the project to prove the effectiveness of the proposed platform.

A preliminary prototype has been successfully tested in an emulation framework. Future works will focus on the performance evaluation of the proposed virtual smart device and on the comparison with commonly employed approaches based on service placement within remote and centralized data centers.

ACKNOWLEDGMENT

This work has been partially supported by the H2020 INPUT project.

REFERENCES

[1] J. Sherry, *et al.*, "Rollback-Recovery for Middleboxes," in Proc. of *SIGCOMM Comput. Commun.*, Rev. 45, vol. 4, August 2015.

[2] J. Sherry and S. Ratnasamy, "A Survey of Enterprise Middlebox Deployments," Technical report, EECS, UC Berkeley, 2012.

[3] R. Morris, *et al.*, "The Click modular router," Proc. of the 17th ACM Symposium on Operating Systems Principles (SOSP '99), Kiawah Island, South Carolina, December 1999.

[4] G. Calarco, *et al.*, "Comparative Analysis of SMP Click Scheduling Techniques," Proc. of QoSIP 2005, Catania (Italy), February 2-4, 2005.

[5] J. Martins *et al.*, "Clickos and the art of network function virtualization," in Proc. of USENIX NSDI, Seattle, WA, April 2-4, 2014.

[6] A. Galis *et al*., "Softwarization of Future Networks and Services -Programmable Enabled Networks as Next Generation Software Defined Networks," *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, 2013.

[7] White paper on "Software-Defined Networking: The New Norm for Networks", https://www.opennetworking.org/.

[8] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proc. of the IEEE, vol.103, no.1, pp.14-76, Jan. 2015.

[9] White paper on "Network Functions Virtualization", http://portal.etsi.org/NFV/NFV_White_Paper.pdf, last access on June 2016.

[10] Cui, Chunfeng *et al.*, " Network Functions Virtualisation (NFV)," in SDN & OpenFlow World Congress, Düsseldorf, 2014, https://portal.etsi.org/portals/0/tbpages/nfv/docs/nfv_white_paper3.pdf

[11] M. Deng, H. Tian, X. Lyu, "Adaptive sequential offloading game for multi-cell Mobile Edge Computing," in Proc. of the 23rd ICT 2016.

[12] G. Maraviglia, *et al.*, "Synchronous Multipoint E-Learning Realized on an Intelligent Software-Router Platform over Unicast Networks: Design and Performance Issues," Proc. ETFA 2007, Patras, Grece, September 25-28, 2007.

[13] A. Lombardo, F. Licandro, G. Schembra, "Multipath Routing and Rate-Controlled Video Encoding in Wireless Video Surveillance Networks," Multimedia Systems, 14, 3, 2008.

[14] R. Y. Chang and C. L. Lee, "IP Video Surveillance Applications over WiMAX Wireless Broadband Technology," 2009 Fifth International Joint Conference on INC, IMS and IDC, Seoul, 2009, pp. 2100-2102.

[15] The INPUT Project, http://www.input-project.eu.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[17] A. G. Busà, *et al.*, "CLAPS: A Cross-Layer Analysis Platform for P2P video Streaming," Proc. IEEE ICC 2007, GLASGOW, Scotland (UK), 24-28 June 2007.

[18] E. C. d. Almeida, G. Sunyé, Y. L. Traon and P. Valduriez, "A Framework for Testing Peer-to-Peer Systems," ISSRE 2008, Seattle, WA, 2008.

[19] M. Barbera, A. Lombardo, G. Schembra, M. Tribastone, "A Markov Model of a Freerider in a BitTorrent P2P Network," Proc. IEEE Globecom 2005, St. Louis, MO, USA, 28 Nov. – 2 Dec. 2005, pp. 985-989.

[20] A. Lombardo, *et al.*, "Measuring and modeling Energy Consumption to design a Green NetFPGA Giga-Router," in Proc. of IEEE Globecom 2012, Anaheim, California, USA, 3-7 December 2012.

[21] K. Hinton, J. Baliga, R. Ayre and R. S. Tucker, "The future Internet - An energy consumption perspective," *2009 14th OptoElectronics and Comm. Conference*, Hong Kong, 2009.

[22] ETSI GS NFV 001, "Network Functions Virtualization (NFV): Use Cases", 10-2013, www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf.

[23] DLNA standard, http://www.dlna.org.

[24] A. Lombardo, *et al.*, "Active Window Management: an efficient gateway mechanism for TCP traffic control", Proc. IEEE ICC 2007, Glasgow, Scotland (UK), 24-28 June 2007.

[25] A. Lombardo, G. Schembra, "Performance evaluation of an Adaptive-Rate MPEG encoder matching IntServ Traffic Constraints," *IEEE Trans. on Networking*, vol. 11, no. 1, pp. 47-65, February 2003.

[26] Y. Sun, Z. Feng and R. R. Ginnavaram, "Non-buffered rate control for real time video compression," *2014 IEEE Global Communications Conference*, Austin, TX, 2014.

[27] L. Galluccio, *et al.*, "An analytical framework for the design of intelligent algorithms for adaptive-rate MPEG video encoding in next generation time-varying wireless networks," *IEEE JSAC*, vol. 23 No. 2, February 2005.

[28] VDI library: www.dieei.unict.it/users/arti/vSTB/vdi.tar.gz

[29] Platinum UPnP, Available online at: https://www.plutinosoft.com/platinum/

[30] Cling - UPnP/DLNA solution for Java, Available online at: https://www.openhub.net/p/cling-upnp

[31] DMC library: www.dieei.unict.it/users/arti/vSTB/DMC.zip

[32] uShare, Available online at: https://wiki.archlinux.org/index.php/UShare

[33] PA library: www.dieei.unict.it/users/arti/vSTB/pa.tar.gz

[34] Multicat, http://www.videolan.org/projects/multicat.html

[35] EA library: www.dieei.unict.it/users/arti/ vSTB/ea.tar.gz

[36] Meet Android Studio, Available online at: http://developer.android.com/tools/studio/index.html

[37] Vitamio SDK, https://www.vitamio.org/en/

[38] Mininet, Available online at: http://mininet.org/

[39] Openflow,www.opennetworking.org/sdn-resources/openflow

[40] OpenDaylight: http://www.opendaylight.org

[41] Xcore LTE EPC Net. Emulator, http://www.accuver.com