A Cloud-based Live Streaming Service for SDN-NFV Enabled Carriers

A. Casella, A. Lombardo*, M. Melita**, S. Micalizzi*, C. Rametta*, G. Schembra*, A. Vassallo***

*** DIEEI, University of Catania, Italy - Email: name.surname@dieei.unict.it**
**** Strategy & Innovation, Joint Open Lab – WAVE, Telecom Italia - Email: name.surname@telecomitalia.it**

ABSTRACT

In the last few years, Software Defined Networks (SDN) and Network Functions Virtualization (NFV) have been introduced in the telecommunications networks as a new way to design, deploy and manage networking services. Working together, they are able to consolidate and deliver the networking components using standard IT virtualization technologies making, in such a way, Telco infrastructures more flexible and adaptive with respect to the needs of both end users and service providers.

In this context, this paper presents a prototype of softwarized live streaming platform allowing carriers to simplify functions/services management and deployment. Target of the proposed framework is enabling small/medium and unusual content providers to share events with their followers without the need of adopting a dedicated and expensive data delivery infrastructure.

CCS Concepts

• Network Services → Cloud Computing; • Network Services → Programmable Networks; • Network Performance Evaluation → Network Experimentation.

Keywords

Software Defined Networking; Network Functions Virtualization; Cloud Computing, Network Orchestration; Media Streaming; Proof-of-Concept.

1. INTRODUCTION

The innovative paradigms of Software Defined Networks (SDN) [1][2] and Network Functions Virtualization (NFV) [3][4][5] have recently redefined the vision of the communications networking, providing network managers with a complete and programmatic control of a dynamic view of the network. The power of SDN is based on its characteristic of decoupling control and data planes, moving the network intelligence to a centralized controller. On the other hand, the emerging technology of NFV introduces an important change in the network service provisioning approach, leveraging IT virtualization technologies to consolidate many network equipment facilities onto standard servers that could be located in data centers, network nodes and even in the end user premises [6][7][8][9][10][11]. Moreover, with the NFV paradigm, network functions become software applications that can easily be migrated according to specific policies aimed at optimizing energy efficiency, costs and performance.

The combined application of both SDN and NFV is strongly stimulating the interest of Service Providers and Network Operators to make the innovation cycles of networks and services faster and easier, reducing both OPEX (operational expenses) and CAPEX (capital expenses), thanks to the enormous possibilities of making operation processes (e.g. configuration of network devices) automatic, and network functions and services more flexible and cheaper. In fact, decoupling network functions, e.g. middle-box functionalities from dedicated hardware devices, and putting them into Virtual Machines (VMs), NFV strongly simplifies functions and services deployment over the network. Moreover, using SDN, traffic control and management functions, like routing, can be moved out of the network nodes and placed on a centralized controller software. Coupling both the approaches, a centralized entity, called Orchestrator, has a complete view of the network allowing it to manage the Control Plane and deploy Virtual Network Functions (VNFs) [12][13]. This represents an enabling factor for a rapid evolution of the dynamic service chain provisioning. Many scenarios of SDN and NFV can be deployed [14][15][16][17][18][19], depending on the network segments (e.g., core or edge) and, consequently, on the exploitation time horizon (e.g., short-, medium- or long-term). Among them, one of the most appealing use cases for Telco Operators and Service Providers is the so-called Virtual Network Function as a Service (VNFaaS). It aims at virtualizing functions of both Customer Premises Equipment (CPE) devices and Provider Edge or Core (PE or PC) nodes, like for example routing, QoS support and Bandwidth differentiation, New-Generation Firewall (NG-FW) and WAN Optimization Control (WOC). According to the general approach of NFV, and thanks to both the growing potential of virtualization frameworks (nowadays more flexible, effective and efficient) and the capability to dynamically and adaptively connect physical and virtual devices according to the SDN paradigm, any application-level service can be virtualized, like for example video encryption [20][21], video mosaic and video compression [22][23][24].

In this context, authors proposed the NetFATE (Network Function At the Edge) architecture [25], whose elements are characterized by fast and simplified deployment with the aim of reducing management costs. Starting from the idea of a fully configurable network, from both the networking and the services points of view, the target of this paper is to propose the idea of a cloud architecture for live streaming broadcasting services with the aim of enabling small/medium and unusual content providers to share events with a restricted number of interested users without the need of adopting a dedicated and expensive data delivery infrastructure and/or subscribing expensive contracts with carriers.

The main goal of this paper is to illustrate a preliminary prototype and the related testbed of the proposed architecture; the whole platform has been realized by using free and open source software and C and Java routines made by ourselves.

More in detail, Section 2 will give a brief description of the

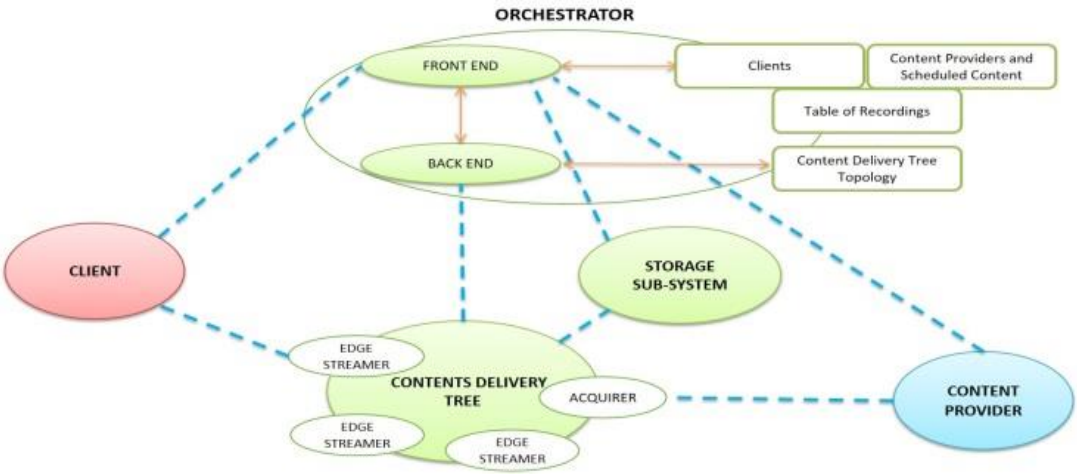

Figure 1 – Overview of the proposed architecture

proposed architecture; in Section 3 we describe the prototype and how CDN elements have been realized; Section 4 presents the prototype testbed; finally, in Section 5 conclusions and future works will be illustrated.

2. REFERENCE ARCHITECTURE

The proposed SDN/NFV live streaming platform is characterized by six kinds of elements:

1) *Client:* it is the end-user of the system. He connects to the TLC network and requires the service to access multimedia content made available by providers exploiting the platform; client will be able to perform some actions such as *select provider*, *select content*, *play or record live contents* and *view recorded content*. The Telco operator provides them a client software (i.e. Mobile APP, Web Portal and similar) thanks to which it is possible to perform the above-mentioned actions. Based on the number of users connected and requiring content, the Telco operator, by using a softwarized approach, can set up and manage the number of nodes and resources involved in the service, in order to drastically reduce the costs of service, making it available to a great number of end-users and small content providers.

2) *Content Provider*: in such a category we do not include great providers, but small/medium or unusual ones such as universities, conference and workshop organizing committee, small municipalities, sports club, training institutions and so on willing to share events with a restricted number of interested users and, for this reason, not having the economic and/or technical capability to adopt a dedicated data delivery infrastructure.

3) *Orchestrator*: it runs on a dedicated server and communicates with all the platform nodes through the Telco operator IP network. Its goal is to allocate, migrate and terminate VMs that execute network/application services, and controlling the traffic paths according to the run-time evolution of the network. More specifically, it is in charge of managing and orchestrating the whole platform, the service chains and the traffic paths according to the amount of traffic crossing the network, the requirements of the Telco operator and the Service Level Agreement (SLA) with the customers. It mainly provides three functionalities:
   a. *Software Defined Network Controller*: it is the core application of a softwarized network, acting as a sort of operating system for the network that sends rules to the nodes about port forwarding, and chooses optimal paths for application data delivery;
   b. *Network Function Virtualization Coordinator*: it refers to the capability of managing the lifecycle of virtual machines running streaming, forwarding and/or storage functionalities inside the TLC network. It is based on the information coming from the Orchestration Engine and the SDN Controller. The NFV Coordinator is in charge of creating, migrating and destroying VMs, communicating with the hypervisors of the network nodes to send them the instructions to manage VMs;
   c. *Orchestration Engine*: it is the component that gathers information about the network (topology, number of connected clients, required content, availability of network devices implementing NFV, etc.). According to specific algorithms, it decides how to manage the resources of the network devices. The implemented policy can refer to a wide range of aspects: energy consumption, end-to-end delay between source and destination of a data flow, number of hops of the path, overall number of virtual machines active at the same time, quality of service, customers service level agreement, etc.

4) *Content Delivery Tree*: it is the structure, made up of the network nodes, the physical and virtual devices, which is responsible of receiving media streams from content providers and routing them in the best way to end users of the service; with the statement ‘in the best way’ we mean according to a decisional scheme, implemented by the network orchestrator, that could be identified with different criteria:
   a. *Consolidation*, i.e. the minimization of the number of nodes and resources involved in the forwarding procedure of the media streams from the source to the destination;

b. *QoS*, i.e. the end-to-end performance perceived by the end-users willing to enjoy the required content, and strictly depending on the number of intermediate nodes, the number of virtual functions each node must process at the same time, etc.

5) *TELCO Provider Edge (PE) Node*: it is the edge node of the carrier, and is equipped with two or more network interfaces in order to guarantee Internet connectivity and provide connection to the client devices. In the proposed architecture, the PE nodes are equipped with virtualization functionalities, in order to create, destroy and migrate virtual machines implementing a set of available network/application functions, and OpenFlow capabilities to provide a switching stack for the hardware virtualization environment.

6) *Storage Sub-System (SS)*: it is the framework component involved in the storage of multimedia content; in this paper we consider the case in which users require the recording of an event, in such a way the SS will be only involved when there is at least a recording request by one of the platform users.

3. DESCRIPTION OF THE PROTOTYPE

In this section we present a free and open source prototype of a simplified version of the proposed platform, describing hardware and software elements employed to realize our testbed. Although in an embryonic stage, it can be considered as the foundations of a more complex middleware.

In the following, we will introduce each element of the prototype.

3.1 Orchestration Node

This component is the core element of the entire platform. It is logically constituted by the following elements: *Front-end and Back-end*.

The Front-end element is an http server written in Java, aiming at handling the RESTful requests from clients. Application data are stored in a relational database containing the following information:

- Users and, for each user, the set of recorded contents, and the list of future streaming events to be played out or recorded;
- Content Providers;
- Video Content and their scheduling.

Once a client wants to enjoy the service, by means of the developed user app he establishes a connection to the server. As a first action, the Front-end verifies the user identity. The user will be so advised about the various content providers, and the related contents broadcasted by them. The user could furthermore choose to access his own contents previously registered. In this case the Front-end will access the *Table of Recordings* database, in order to retrieve the requested content and forward it to the most suitable Edge Streamer.

The *Front-end* also interacts with the Back-end server that manages the transmission tree.

The Back-end Server has the following three main functions:

1. It is responsible for the communication between the Front-end Server and the Content Delivery tree and for the configuration of this latter; so it is called every time the Front-end:
   - receives a connection request from a client or a content provider;
   - a client requires a new live content and a path between source and destination has to be created;
   - a content provider begins the transmission of a new live event.
2. It checks the state of the network, the usage of a link between two nodes, the RAM/CPU usage of computation nodes and/or virtual machines running network or application functionalities, with the scope of choosing the physical and virtual resource that will serve a specific request or selecting the most efficient end-to-end path between source and destination.
3. It communicates with the virtualization manager in order to create/destroy/migrate and configure virtual machines running networking and application services.

In our prototype, the Back-end server is realized in C language, runs inside a server or a virtual machine, and is connected to all nodes of the content delivery.

3.2 Content Delivery Tree

The delivery tree is composed by three elements:

1) *Acquirer*: it is the access node of the platform for the Content Providers and it represents the root of the content delivery tree; its main tasks are: recognizing the media streams; notifying the Back-end about the upcoming transmissions; receiving transmission instructions (in the form of a list of IP addresses where the media stream have to be forwarded). This node is dynamically instantiated by the Back-end in the Telco Provider Edge node, in order to be as much close as possible to the Content Provider;

2) *Forwarders/Core Network Nodes*: they are the network nodes between the acquirer and the edge streamers, i.e. the TELCO network nodes serving end users. Analogously to the Acquirer, they receive information from the Back-end about next hops (where the media stream has to be forwarded).

3) *Edge Streamers*: they are the edge nodes of the network, and serve the end users of the platform. Like the other delivery-tree elements, they receive instructions from the Back-end. Edge Streamers may expose additional functionalities to the end users, such as transcoding of high quality video to a lower quality video format to fit both end users and networks requirements.

3.3 TELCO Provider Edge Node

PE nodes have been realized through general-purpose computers, equipped with two network interfaces, one providing connectivity to client devices belonging to the local area network, the other connected to the TELCO IP core network. The PE architecture is shown in Figure 2.

The host OS is CentOS release 6.4, directly installed on the bare metal, whereas we have chosen Xen Hypervisor as virtual machines hypervisor, that is the middleware software enabling both virtual system management and hardware resource sharing with the aim of executing multiple computer operating systems on the same hardware concurrently. Xen Hypervisor is a free and open-source software, developed by the University of Cambridge Computer Laboratory, available for x86-64, IA-32 and ARM architectures, supporting a form of virtualization known as para-virtualization. Thanks to the para-virtualization paradigm, it is possible to obtain a faster execution of the VMs as compared with

the traditional approaches. More in detail, Xen Hypervisor permits the virtual machines to share directly the x86-64 hardware without the need to preventively specify the resources to be allocated for the virtualized hardware. Under this perspective, the host OS only acts as a console to operate with the virtualization manager, enabling commands as create, destroy and migrate.

Finally, OpenvSwitch has been installed on the host OS in order to create and configure SDN-compliant virtual switches. Once created, it is possible to connect both real and virtual network interfaces to the virtual switches, and enable the remote control of them by setting up a software-defined controller.

3.4 Clients

Despite the platform is able to provide its functionalities to different kinds of clients, such as web based or mobile clients, in our testbed we developed a mobile Android client app (refer to Figure 3) with the purpose of testing the platform functionalities.

The mobile app has been developed ensuring frontward compatibility with Android 4.4. It allows users to play live video contents transmitted by content providers, where video transmission is based on RTP protocol.

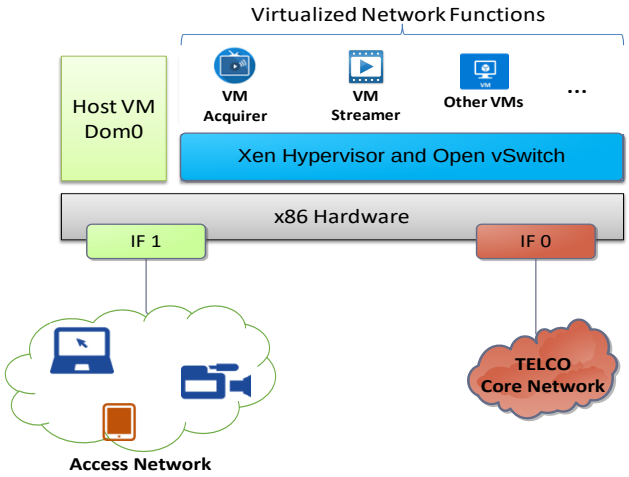

Figure 2 - Architecture of a Provider Edge Node

Users could access to the contents scheduling of each content provider, 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, users are able to record a live event even if it is already started.

This feature enables new scenarios if these actions are collected. In fact it is possible to define a set of metrics and, according with these, the orchestrator can plan the allocation of the relative resources.

The app is composed by three sections:

- Discover, showing the contents providers list and, for each one, the scheduled events.
- MyShare, which shows contents previously recorded, giving the possibility to access and play out them on demand.
- Calendar, a calendar section containing events watched in the past. In addition, it allows the reservation and the notification service related to future scheduled events.

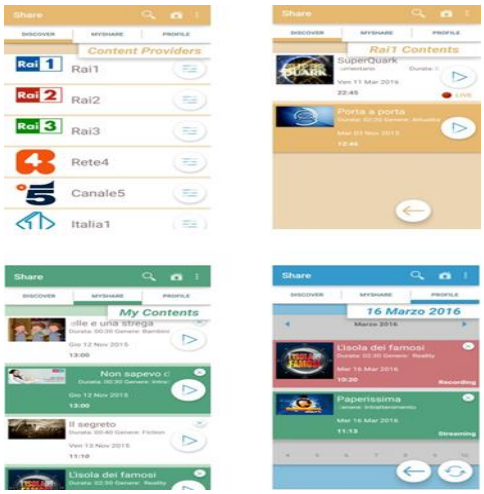

Figure 3 - Some screenshots of the developed User App

3.5 Content Providers

In order to allow Content Providers to expose their own contents, a web application has been released. The main functionalities of this application are:

- Provide a schedule for the future events to be broadcasted with a relative description;
- Streaming a content (either live or registered) to a destination *Acquirer node*, according to the rules set by the back end process, towards the couple IP address-port number advertised by the front-end process and dedicated to each streaming flow).

When the Content Provider accesses the service through his dedicated web app, the Front-end has to access his database, in particular the table regarding the Content Provider and Scheduled contents, in order to synchronize the information among all the database.

4. TESTBED DESCRIPTION AND CASE STUDY

In this section we describe a testbed to evaluate the effectiveness of the proposed architecture. We considered a simple proof-of-concept (PoC) representing a subset of the network topology presented in Figure 4 composed by:

- two smartphones with Android OS 4.4, where the client application has been installed;
- one x86 PC, equipped with a webcam and the CP web application, acting as content provider;

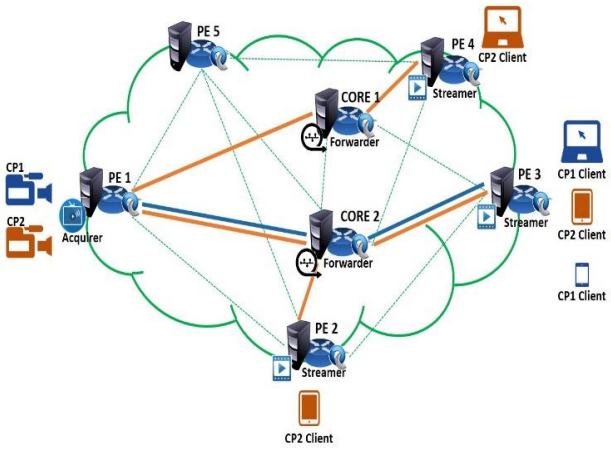

Figure 4. Testbed network topology.

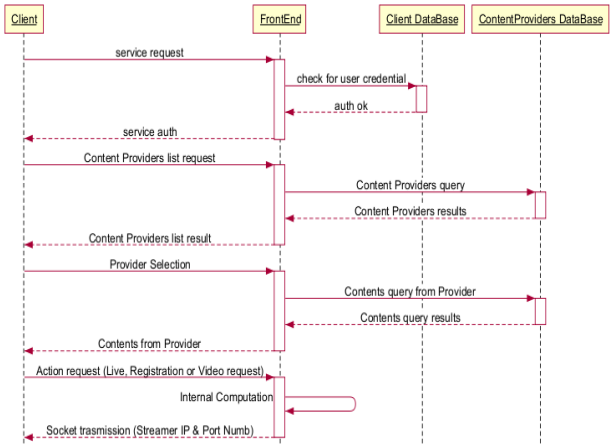

Figure 5 - Message exchange between the User App (client) and the Front-end server

- two provider edge nodes, realized by using x86 general-purpose hardware, equipped with two network interfaces: the first acting as network access point to provide connectivity to the smartphones; the second one employed to connect the provider edge node to the TELCO core network; PE nodes run Xen Hypervisor as virtualization environment for the virtual machines implementing Acquirer and Edge Streamer functionalities, and OpenvSwitch coupled with POX SDN controller to provide virtual network functions;
- one remote server acting as application server, i.e. the above mentioned front-end server;
- one remote server running the back end server, the NFV manager (a C client-server application communicating with Xen Hypervisor) and the POX controller to configure the OpenvSwitch of the PE nodes;
- one workstation virtualizing the other network nodes.

The network topology employed during our testbed consists of five PE nodes, providing network access and connectivity to the client devices, both content providers and end-user devices, and two core nodes acting as routers.

Two content providers have been connected to the platform in order to test the live streaming functionalities.

When a client accesses the network by means of the User App his data traffic is re-routed towards the Front-end server in order to perform the authentication and authorization procedure; once the procedure terminates, end user is able to view the list of content providers and the related schedule in order to select the required content and the action to perform (refer to Figure 5). Let us suppose he chooses the *live view* option. If this is the case, the

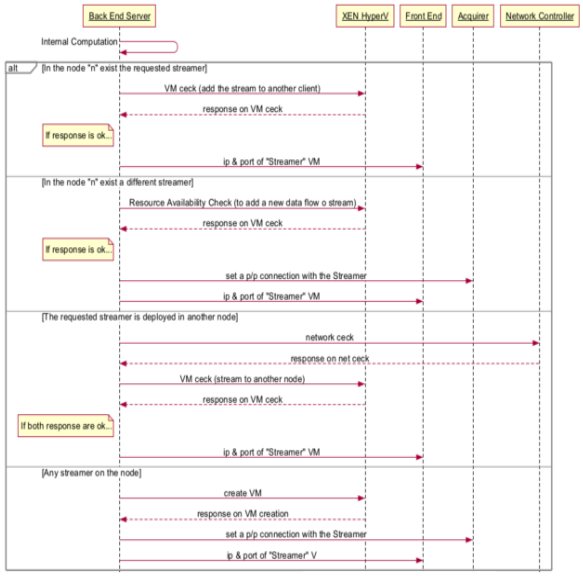

Figure 6 - Back-end server workflow when a live content is required

Front-end server communicates the user's selection to the Back-end server that executes the procedure, as shown in Figure 6 to establish the connection between the content acquirer node and the edge streamer node. More in detail, the Back-end server receives from the Application Server the client ID, the required content, in the following referred to as content X, and the PE ID, that is the identification code of the PE access node where the client is connected to the network. According to the above information, the Back-end server checks if a streamer is running on the PE access node and, if any, two cases may occur:

- if the Edge Streamer VM is already processing the content X, then the Back-end process assigns the Edge Streamer to the client and communicates its IP address to the Front-end server;
- if the running Edge Streamer VM is not processing the content X but it has enough computational and bandwidth resources, the Back-end server assigns the data flow to it and configures the connection between Acquirer and Edge Streamer; then as well as the above mentioned case the Edge Streamer IP address is communicated to the Front end server.

Otherwise, if no Edge Streamer is instantiated on the access PE, the Back-end process checks if a suitable ES is running on a neighbor PE node.

Finally, if it is not possible to assign an existing Edge Streamer, then the Back-end process communicates to the Hypervisor in order to instantiate a new Edge Streamer VM; once running, the Back-end process sets the communication between the Acquirer node and the Edge Streamer.

5. CONCLUSION AND FUTURE WORKS

This paper presents the architecture and a first prototype of a Softwarized Live Streaming platform. The *software defined networking* section has been entirely implemented by using C routines and the *multicat* [26] tool to forward and duplicate incoming multimedia data streams across the nodes of the core network. The *network function virtualization* has been realized by using *Xen Hypervisor* and a proprietary C-based client/server application to create, destroy and manage the virtual machines running the Acquirer and the Edge Streamer functionalities at the edge nodes of the TELCO network.

As a future work we will introduce the storage of on demand contents by using FEC and UnFEC techniques, to enhance the contents retrieval performance perceived by the users, and we will migrate this architecture toward a full SDN network with the goal of substituting the C routines with an OpenFlow-compliant language and a framework such as OpenDaylight [27] or OpenContrail [28].

6. ACKNOWLEDGEMENTS

This work has been performed in collaboration with Telecom Italia Joint Open Lab WAVE, and partially supported by the INPUT (In-Network Programmability for next-generation personal cloUd service supporT) project INPUT [29] funded by the European Commission under the Horizon 2020 Programme (Call H2020-ICT-2014-1, Grant no. 644672).

7. REFERENCES

[1] White paper on “Software-Defined Networking: The New Norm for Networks”, available at *https://www.opennetworking.org/*.

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

[3] White paper on “Network Functions Virtualisation”, av. At *http://portal.etsi.org/NFV/NFV_White_Paper.pdf*.

[4] A. Manzalini et al., “Software-Defined Networks for Future Networks and Services,” White Paper based on the IEEE Workshop SDN4FNS.

[5] Network Functions Virtualisation – White Paper #3, “Network Operator Perspectives on Industry Progress,” available at *https://portal.etsi.org/ Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf*

[6] A. Lombardo, A. Manzalini V. Riccobene, G. Schembra, “An Open Architecture for Software Defined Services at the Edge,” EuCNC2014, Bologna, Italy, June 23/26, 2014.

[7] A. Manzalini and N. Crespi, "An edge operating system enabling anything-as-a-service," in IEEE Communications Magazine, vol. 54, no. 3, pp. 62-67, March 2016.

[8] A. Lombardo, M. Barbera, C. Panarello, G. Schembra, “Active Window Management: an efficient gateway mechanism for TCP traffic control”, Proc. IEEE ICC 2007, GLASGOW, Scotland (UK), 24-28 June 2007.

[9] Q. Yaseen, F. AlBalas, Y. Jararweh and M. Al-Ayyoub, "A Fog Computing Based System for Selective Forwarding Detection in Mobile Wireless Sensor Networks", IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W), Augsburg, Germany, 2016, pp. 256-262.

[10] M. Taneja and A. Davy, "Poster Abstract: Resource Aware Placement of Data Stream Analytics Operators on Fog Infrastructure for Internet of Things Applications," 2016 IEEE/ACM Symposium on Edge Computing (SEC), Washington, DC, USA, 2016, pp. 113-114.

[11] A. Sciarrone, C. Fiandrino, I. Bisio, F. Lavagetto, D. Kliazovich and P. Bouvry, "Smart Probabilistic Fingerprinting for Indoor Localization over Fog Computing Platforms," 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, 2016, pp. 39-44.

[12] ETSI GS NFV 002, “Network Functions Virtualization (NFV): Architectural Framework”, 10-2013, available at *http://www.etsi.org/ deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010 101p.pdf*.

[13] P. Costa, M. Migliavacca, P. Pietzuch and A. L. Wolf, “NaaS: Network-as-a-Service in the Cloud,” in Proc. of Hot-ICE’12, April 24 2012, San Jose, CA.

[14] ETSI GS NFV 001, “Network Functions Virtualization (NFV): Use Cases”, 10-2013, available at *http://www.etsi.org/deliver/etsi_gs/ nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf*.

[15] ETSI – PoC, “NFV Proofs of Concept,” available at *http://www.etsi.org/ technologies-clusters/technologies/nfv/nfv-poc*.

[16] G. Maraviglia, M. Masi, V. Merlo, F. Licandro, A. Russo, G. Schembra, “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.

[17] T. Li, H. Zhou, H. Luo, Q. Xu and Y. Ye, "Using SDN and NFV to Implement Satellite Communication Networks," 2016 International Conference on Networking and Network Applications (NaNA), Hakodate, 2016, pp. 131-134.

[18] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, “Software-defined and virtualized future mobile and wireless networks: A survey,” Mobile Network Appl., vol. 20, no. 1, pp. 4–18, Feb. 2015.

[19] A. Lombardo, F. Licandro, G. Schembra, “Multipath Routing and Rate-Controlled Video Encoding in Wireless Video Surveillance Networks,” Multimedia Systems, Volume 14, Number 3, 155-165.

[20] T. P. Pai, M. E. Raghu and K. C. Ravishankar, "Video Encryption for Secure Multimedia Transmission - A Layered

Approach," 2014 3rd International Conference on Eco-friendly Computing and Communication Systems, Mangalore, 2014, pp. 127-132.

[21] D. M. Dumbere and N. J. Janwe, "Video encryption using AES algorithm," Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014, Coimbatore, 2014, pp. 332-337.

[22] A. Lombardo, G. Morabito, G. Schembra, "Modeling Intramedia and Intermedia Relationships in Multimedia Network Analysis through Multiple Time-scale statistics," IEEE Transactions on Multimedia, vol. 6, no. 1, pp. 142-157, February 2004.

[23] A. Cernuto, F. Cocimano, A. Lombardo, G. Schembra "A Queueing System Model for the Design of Feedback Laws in Rate-Controlled MPEG Video Encoders," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 4, pp. 238-255, April 2002.

[24] H. Chen, B. Zhao and Y. Liu, "A Video Mosaic Algorithm for Compressed Video," Second Workshop on Digital Media and its Application in Museum & Heritages (DMAMH 2007), Chongqing, 2007, pp. 370-377.

[25] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, V. Riccobene, "An open framework to enable NetFATE (Network Functions at the edge)", in Proc. of 1st IEEE Conference on Network Softwarization (NetSoft), 2015, 13-17 April 2015.

[26] Multicat tool, available at: http://www.videolan.org/projects/multicat.html

[27] OpenDaylight, "OpenDaylight: A Linux Foundation Collaborative Project," 2013. [Online]. Available: http://www.opendaylight.org

[28] Juniper Networks, "Opencontrail," 2013. [Online]. Available: http://opencontrail.org/

[29] INPUT Project website available at http://www.input-project.eu