On-demand, Dynamic and at-the-Edge VNF Deployment Model Application to Web Real-Time Communications

Amina Boubendir* †, Emmanuel Bertin‡ and Noëmie Simoni†

*Convergent Network Control Architectures Department, Orange Labs, Chatillon, France
†Networking and Computer Science Department, Télécom ParisTech, Paris, France
‡Unified Communications and Advanced Services Department, Orange Labs, Caen, France
* ‡ {amina.boubendir, emmanuel.bertin}@orange.com, †noemie.simoni@telecom-paristech.fr

***Abstract*—In the context of Network-as-a-Service (NaaS) for network operators openness towards Over-The-Top (OTTs) players, we propose an *"On-demand and Dynamic Deployment Model for Virtual Network Functions (VNFs)"*. We consider both, Telco's needs for more dynamicity in network service delivery, and OTTs' needs for network functions that support requirements of applications at the network-level. In order to allow third-party applications to request deployment of VNFs for specific needs, the proposed deployment model is based on an exposition approach that uses Network APIs over Network Functions Virtualization (NFV) Management and Orchestration modules. To illustrate the proposed model, we focus on Web Real-Time Communications (WebRTC). WebRTC applications are versatile and have strong needs for network functions which makes them a relevant use-case. We have implemented the proposed approach on *OPNFV* platform. We exploit the "at-the-edge" feature and location-awareness to respect the time-sensitivity and QoS constraints of WebRTC communications. For faster VNF instantiation, we rely on container-based virtualization *(Docker containers)*. Finally, we perform experiments and evaluate the advantages of this model comparing to legacy and alternative approaches.**

***Index Terms*—NFV, Dynamicity, On demand Deployment, Network APIs, Edge Computing, Real-Time, Telcos, OTTs.**

I. Introduction & Motivations

With the evolution of Over-The-Top (OTT) services (Cloud, Content and real-time applications), users expect ubiquitous access, via any access network, with strong mobility and high requirements of network-level Quality of Service (QoS) and Quality of Experience (QoE). It became clear that these applications evolve faster than networks and reshape application-to-network communication models which raises challenges in operators networks. To answer applications needs, network operators openness aims at providing access to value-added and network-level services using exposition of Telco network functions through Network Application Programming Interfaces (APIs). However, exposed network functions are legacy specific physical nodes, designed and deployed over legacy dedicated network infrastructures with long deployment cycles and slow Time-To-Market and the used APIs are (vendor) specific APIs and not Open. Of course, this is not adapted to today's users and applications who have continuously changing requirements and very versatile behaviors that require immediate network-level reactions. In legacy exposition approaches, the static nature of networks makes it difficult for operators to offer dynamically adapted and customized network services and even more difficult to deploy a network function on-demand for an application.

Our objective is to promote Telco openness with efficient exposition and on-demand deployment of network functions. In front of these needs for more dynamicity in networks, we rely on Network Functions Virtualization (NFV) [1] and Open APIs as strong enablers. With this article, we complete our work on Network-as-a-Service (NaaS) architecture presented in [2]. Therefore, we propose a dynamic deployment model that allows third-party applications and service providers to benefit from a differentiated orchestration of Virtual Network Functions (VNFs). To achieve our model for dynamic interactions between OTTs and Telcos, we introduce on-demand and dynamicity features as the results of the convergence of control plane and management plane in networks. For that, we propose to expose Network APIs rather by NFV Management and Orchestration building blocks [3] to allow VNF discovery, selection then a location-aware deployment. Moreover, we rely on Telco assets that cope with Edge Computing promises [4] to deploy functions at network edge, close to end-users for fast request handling, reduced delays and improved QoS. Furthermore, we rely on container-based virtualization which is becoming a pillar within NFV to accelerate VNFs deployment, especially for real-time applications.

In the next Section, we survey related work. Then, in Section III, we present our proposition of on-demand, dynamic deployment model and highlight its features. Later, Section IV describes in details the implementation of our proposition and the illustrating WebRTC use-case. Finally, Section V, presents and discusses the evaluation results. This leads us to give our conclusions and future work.

II. Related Work

To position our paper, we survey works related to: dynamic deployment of network functions, the exposition of Telco network functions and Edge computing in NFV.

A. Dynamic Deployment of Network Functions

T-NOVA project proposes a framework for dynamically providing VNF as a service (VNFaaS) [5] to allow operators to deploy distinct VNFs to be provided on-demand as value-added services, [6] [7]. The solutions focus on business-related and customer front-end aspects.

That is how the stakeholders interact with service but mostly on resource-aware scheduling methods to ensure optimal use of resources. Also, Open Networking Foundation (ONF) has proposed a solution for deployment in NFV in order to deal with the dynamic provisioning of network services [8], and a solution for Operator Network Monetization [9] promoting linkage between the operator's network and the applications. However, these works do not integrate a VNF selection and discovery process using network APIs. Moreover, even dealing with the same network-to-application visibility problem, these solutions are only networking-oriented.

B. *Network APIs for Network Functions Exposition*

In [10], Open Mobile Alliance (OMA) has provided sets of specifications for Network APIs. These APIs are between the service access layer and network capabilities to allow operators to expose fundamental capabilities such as Location Services and any core network function (e.g., Identity, Messaging, Billing and Payment, QoS, and Enhanced Communications) in an open and programmable way. Also, the work in [11] presents a solution of opening legacy network functions in 3G mobile architecture via a QoS control API to third parties. We have analyzed the proposed Network APIs to understand the general approach used today:

1) OMA QoS Network API: allows to request changing a QoS level by interfacing with Evolved Packet Core (EPC) functions and Policy and Charging Rules Function via Rx 3GPP interface through a gateway.

2) OMA RCS Network API: allows to built RCS-based services using a gateway in operator network to manage interworking between server API and RCS/IP Multimedia Subsystem (IMS) network functions.

We illustrate the exposition model of the legacy approach in Figure 1. Based on our analysis, existing network functions are exposed through (vendor) specific APIs. They are implemented based on an approach that requires first to deploy a gateway in operator network. As legacy architectures are heterogeneous and physical network functions are configurable (with opposition to programmable), different protocols are needed and used. These gateways are needed to act as intermediary nodes to interface with existing physical network functions.

C. *Edge Computing for NFV*

The work in [12] is a relevant reference for our work. It presents an NFV edge-cloud framework to enable operators to realize any service through open-APIs. It discusses the functional components in overlay deployment in NFV while we consider the on-demand deployment.

III. On-demand, Dynamic and at-The-Edge Virtual Network Functions Deployment Model

In this Section, we present our propositions for a VNF deployment model and an exposition approach. We describe the on-demand and dynamic deployment of VNFs at the Telco Edge Cloud and highlight the features of the proposed model with the brought advantages.

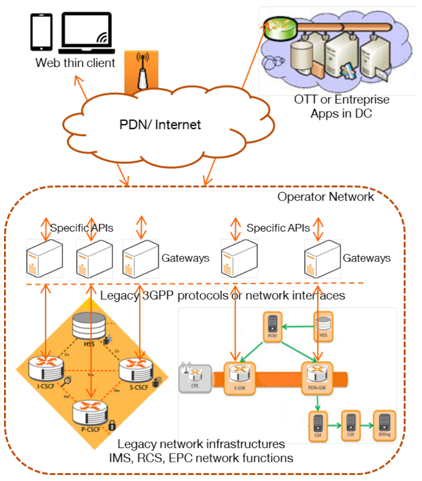
Web thin client
OTT or Entreprise Apps in DC
PDN/ Internet
Operator Network
Specific APIs
Specific APIs
Gateways
Gateways
Legacy 3GPP protocols or network interfaces
HSS
I-CSCF
S-CSCF
P-CSCF
Legacy network infrastructures IMS, RCS, EPC network functions

Fig. 1: Legacy approach of Physical Network Functions Exposition using Specific APIs and Gateways

A. *Description of the Proposed Deployment Model*

It is important to recall that it is a follow-up work to the Network-as-a-Service (NaaS) architecture presented in [2]. In order to allow third-party applications to benefit from on-demand deployment and differentiated orchestration of network services, we propose a deployment model using a new exposition approach. We propose that Network APIs be exposed by network applications included in NFV management and orchestration building blocks, namely NFV Orchestrator (NFVO). This is because we define the dynamicity feature as the result from the convergence of control and management planes. Indeed, the management plane is the one capable of considering user requests with temporal and spatial preferences and integrating them in data plane service delivery. While on the control (signaling) plane, OTT applications will use such exposed Network APIs to express users needs and preferences and to request the deployment of specific VNFs from operator management plane. Upon a request through the Network API, NFV orchestrator asks VNF Manager (VNFM), in charge of placement and life-cycle management, to deploy the requested VNF. Then, the VNFM uses VNF deployment templates to ask Virtual Infrastructure Manager (VIM) to instantiate the VNF over the NFV Infrastructure (NFVI). Figure 2 synthesizes our propositions. We further describe the proposed model through its features in details.

B. *Features of the Proposed Deployment Model*

There are three main features: “on-demand”, “dynamic” and “at-the-edge” features. The on-demand and dynamic features, for both the demand and the deployment, are achieved thanks to the convergence of control and management planes that relies on Open Network APIs, and thanks to the automation in deployment by NFV. The at-the-edge feature is achieved thanks to edge computing, location-awareness and to the Telco extended footprint.

1) On-demand Deployment Feature: Physical network functions are today pre-deployed based on algorithms and distribution or location of users which requires network dimensioning and congestion control mechanisms for requests arrivals. They are usually over-dimensioned which induces CAPEX and OPEX and often long deployment cycles. Unlike this classical deployment of network functions, the on-demand deployment allows activating a network function or a service only when there is a request, i.e., we are sure that there are users needing the service. This feature is coupled with Dynamic Network APIs to allow a discovery and selection using the Open Network API.

2) Dynamic Deployment Feature: The dynamic feature is enabled by *the convergence of control and management planes*. First, the dynamicity in demand relies on Network APIs. The proposed model allows dynamicity in interactions between application-layer and network layer using easy-to-use and easy-to-implement REST Network APIs [13]. REST is a software architectural pattern aiming to simply describe an API using HTTP client-server protocols without defining extra-layers. REST design is a prerequisite for adoption by application developers community. Second, the dynamicity in deployment relies on NFV automation but we leverage this feature using NFV with container-based virtualization approaches [14] which allows a dynamic deployment time reduced from several minutes to seconds comparing to VMs.

3) At-the-Edge Deployment Feature: The at-the-edge feature is achieved through the use of information about users location in the network using geo-localization APIs in users applications in order to choose the best NFVI PoP location for VNF deployment at the network edge or in edge clouds. Comparing to a deployment on OTT Cloud, on Telco Cloud or on core network, processing of network functions occurs at the network edge, rather than completely in the cloud. As edge computing addresses network concerns [15], this features allows to achieve network resource (bandwidth) savings and to benefit from delay reduction for network traversal as the client and server are at a one-hop away distance, and thus offer better QoS and QoE. The use of container-based approach is also an enabler for this feature as Docker has been evaluated as a viable Edge Computing platform [16].

We have implemented the proposed deployment model and then instantiated it with a WebRTC service (Section IV). The implementation of these features cross-answer QoS constraints and VNF deployment issues as we show in Section V.

IV. Implementation & Illustrating Case of Study On-demand VNF deployment for WebRTC

A. Implementation of the Proposed Deployment Model

The implementation of the proposed deployment model is illustrated in Figure 3. The right half side of the figure shows the implemented NFV management modules. We have relied in this implementation on the OPNFV platform [17] which represents the NFV Infrastructure. The NFVI resources are located in an NFVI-PoP at the edge of the network. Actually, we have two edge NFVI-PoPs (not shown in the figure).

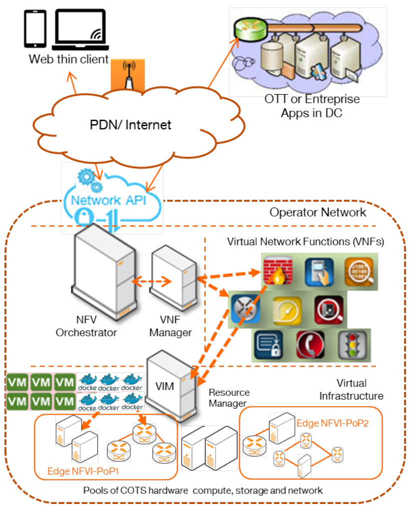

Fig. 2: Proposed Deployment Model: NFV-based VNF Exposition and On-demand Dynamic Deployment on Telco Edge

A node based in *Paris* and a node based in *Lannion*. We have used OpenStack infrastructure orchestrator representing the VIM to manage the set of virtual resources (server stacks for compute and storage and switches for networking) comprised in each NFVI-PoP. These resources are connected to the public WAN, i.e., VM instances have public IP addresses. An Opendaylight SDN controller is also installed to program the switches of the NFVI. For faster instantiation of VNFs, we used container-based virtualization approach and precisely Docker [14] instead of VMs as the Docker software packages include all elements required to execute VNF container since the needed libraries are pre-installed in the VNF docker image. We have considered Cloudify platform manager to represent the NFV Orchestrator and the VNF Manager in our implementation. It is responsible for the deployment using “.yaml” deployment templates and for the management of VNF life-cycle (starting, terminating,...etc). As the proposed model suggests, in order to allow interactions between an OTT application and Telco NFV management blocks this latter has to expose Open Network APIs to offer deployment services.

We have then developed a REST Network API for VNF deployment request exposed through Cloudify (NFVO + VNFM) module. The proposed Network API is a RESTful API written in python based on the OMA guidelines for JSON/RESTful Network APIs [18]. It receives a “GET” request: a VNF deployment or allocation request which provokes a VNF instance creation and sends response, in a JSON format, to the application including the VNF virtual machine IP address and port number, the VNF container IP address and port number or the VNF container ID. We have instantiated the proposed approach with a WebRTC communication service as shown in the left half side of Figure 3. We will present this illustration in the following. Let us first give a brief overview of WebRTC services and their shortcomings which highlights their network-level needs.

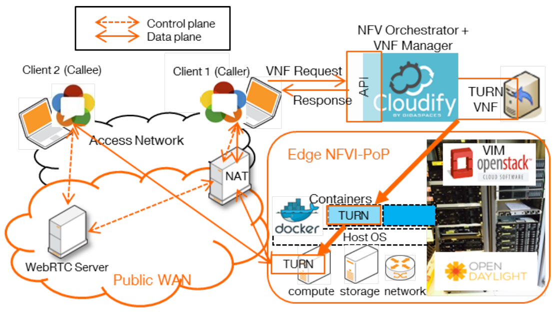

Fig. 3: Implementation of the proposed approach

B. *WebRTC Services Overview & Shortcomings*

1) Overview: WebRTC [19] is a recent technology driven by web actors. It allows versatile browser-to-browser audio and video communications, screen sharing and data exchange. Typically a WebRTC communication between two peers is established first through a *PeerConnection* signaling request from a WebRTC Client to a WebRTC Server on the control plane, then directly between two WebRTC peers on the media plane once the connection is established. Ideally, there should be no intermediary network functions or media relays in the media plane between peers as this may add network delays.

2) Shortcomings & Network-level needs: Actually, media relays are so needed for WebRTC services because of network architecture implementations. In reality most end users (peers) are behind one or more layers of Network Address Translation (NAT), some have anti-virus software that blocks certain ports and protocols, and many are behind proxies and corporate firewalls, so peers have non-routable IP addresses and thus P2P networking cannot be achieved. In such cases, Interactive Connectivity Establishment (ICE) mechanism is needed to gather candidates of STUN servers (Session Traversal Utilities for NAT), to get an external network address, and TURN servers (Traversal Using Relays around NAT) to solve more difficult NAT traversal situations (in case of symmetric NAT for example). TURN servers are needed to relay traffic if direct (peer to peer) connection fails. Invoking TURN always returns connection success as every TURN server supports STUN. A TURN server is a STUN server with added networking and relaying functionality built in. TURN allocates a public relayed IP address for the connection. The WebRTC client advertises this TURN address to be reached then all media data is relayed between peers via TURN. This makes TURN server a crucial and value-added network function that can be deployed to ensure WebRTC communications connection success. Hence, we take TURN server as an example of VNF to be requested for on-demand deployment.

C. *On-demand TURN for Web Real-Time Communications: Illustration Scenario*

We have considered a WebRTC application to illustrate the proposed model of Figure 2 as shown on Figure 3.

1) Used Tools and Settings: We have chosen *AppRTC* as the WebRTC application running a on web browser.

Coturn is the TURN server code which represents the VNF to be requested for deployment. We have used two laptops for a caller and a callee on the same wireless network both are behind an enterprise NAT using one webRTC server.

2) Scenario Description: The global scenario is about a WebRTC video conferencing application that requests a TURN server as a VNF from a network operator using the on-demand deployment model and the proposed Network API exposed by Cloudify NFV orchestrator. Starting from the WebRTC browser to browser call flow specified by W3C , we propose the scenario summarized by the call flow of Figure 4 to show the on-demand dynamic deployment of a TURN VNF based on a demand coming from the WebRTC application at the beginning of the session. Only the caller side is represented but the procedure is the same for the callee. Step-by-step, the call flow shows the initialization of a WebRTC call session using the proposed approach.

V. Experiments & Evaluation Results

We describe in this Section the conducted experiments and present our results to evaluate the proposed approach, then discuss the evaluation results to highlight the advantages.

A. *Experiments*

In addition to the legacy approach where network functions are already deployed as physical nodes, we have considered other situations regarding the VNF and the resource infrastructure provider (being a network operator, an OTT or a Cloud service provider) and different possible deployment locations (Network, OTT Cloud, Telco Cloud or Edge Cloud). So, we have defined the following experimentation scenarios for our use-case. For each scenario, we consider a WebRTC video communication and vary conditions combinating these factors.

1) TURN as a Physical Network Function in the WAN: This scenario represents the legacy approach for providing a network function. We use a TURN server deployed as a physical network function deployed over the WAN, not at the network edge or near access networks. Two possibilities have been used: physical TURN deployed by a Telco in the WAN and physical TURN deployed by an OTT CSP in the Internet.

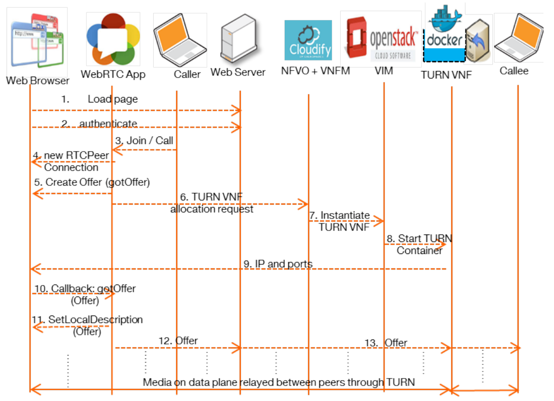
Web Browser
WebRTC App
Caller
Web Server
NFVO + VNFM
VIM
TURN VNF
Callee
1. Load page
2. authenticate
3. Join / Call
4. new RTCPeer Connection
5. Create Offer (gotOffer)
6. TURN VNF allocation request
7. Instantiate TURN VNF
8. Start TURN Container
9. IP and ports
10. Callback: gotOffer (Offer)
11. SetLocalDescription (Offer)
12. Offer
13. Offer
Media on data plane relayed between peers through TURN

Fig. 4: On-demand TURN VNF deployment call flow

Thus, in the experiments of this scenario, we configure the two WebRTC clients to use the physical TURN server.

2) TURN as a Virtual Network Function in OTT Cloud: We deploy TURN as a VNF in a VM over the infrastructure (IaaS) of an OTT Public Cloud provider. In this scenario's experiments, we configure the two WebRTC clients to use this TURN VNF (VM). This is done somehow using the legacy approach since the TURN VNF is pre-deployed.

3) TURN as a Virtual Network Function in Telco Cloud: We deploy TURN as a VNF in a VM over the infrastructure (IaaS) of a Telco Cloud at a regional area. Thus, in the experiments of this scenario, we configure the two WebRTC clients to use this TURN VNF (VM). Here also, legacy approach is used since the TURN VNF is pre-deployed.

4) TURN as a Virtual Network Function in Telco Edge Cloud: We use the proposed approach of on-demand, at-the-edge deployment. The TURN VNF is not pre-deployed and the WebRTC clients are not pre-configured, they request the deployment of a "TURN VNF container" on-the-fly. Based on their location, TURN is instantiated over Telco Edge Cloud.

5) Direct Peer-to-Peer WebRTC communication: As a reference or objective for our measurements, we have considered a direct P2P WebRTC communication without going through a TURN media relay as it is the ideal situation.

B. Measurements

We have measured call-related QoS core performance metrics [20]. We have made 10 experiments per scenario with 20 measurements per experiment for each metric.

- **Total time for call setup** caller point of view, in seconds. A very important metric to evaluate the time needed to establish a call using our approach, i.e. including the request through the Network API and VNF deployment .
- **End-to-end delay**, related to one-way delay and Round-Trip Time (in ms). To characterize the relevant end-to-end delays, we use the Inter Quartile Range (IQR), i.e. the values where 25% and 75% of the measurements lie.
- **Jitter, the packet inter-arrival delay variation** related to the one-way delay (in ms). Here, the monitor is at the callee as we assume the caller is the sender. Jitter dD, is $dD_i = D_i - D_{i-1}$ where D_i is the measured one-way delay for measurement i. We also use the IQR.

C. Evaluation Results

We present and discuss our evaluation results per metric for each scenario. We used Matlab *Boxplot* tool as it gives relevant information regarding values distribution and their variation around a mean value represented by a target point for each experiment. The X axis represents the experiment number.

1) Call Setup Time: Figure 5 gives the variation of video call setup time in seconds. The results of scenarios 1 and 2 are more or less comparable but not stable. Results of scenario 3 are more satisfying as they are more stable and call setup time is reduced to a mean time of 6.5s. Using the proposed approach (scenario 4), call setup time is further reduced closest from results of scenario 5 (ideal).

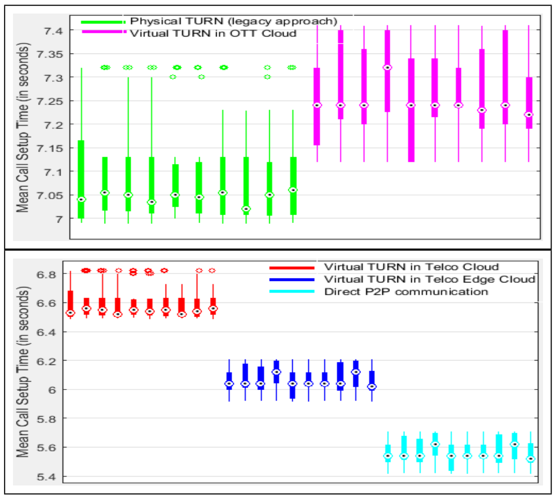

Fig. 5: Evaluation of Call Setup Time Variation

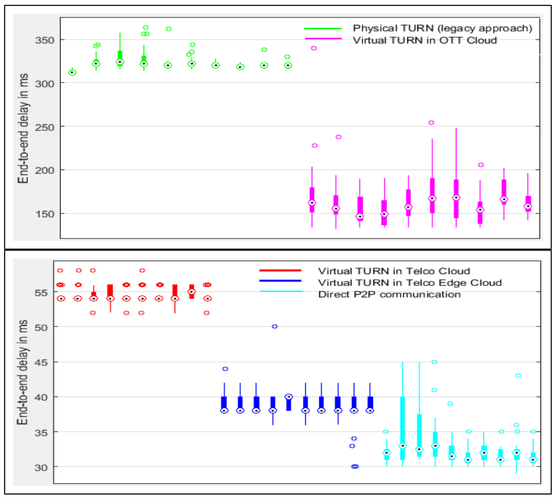

Fig. 6: Evaluation of End-to-End Delay Distribution

It is important to mention that this scenario uses the proposed solution which means that the measured call setup time includes the time for sending request and receiving response through the Network API and the time to deploy the TURN VNF container at the Telco Edge Cloud. These results show that the call setup time using the on-demand dynamic deployment model is almost as fast as P2P call setup time.

2) End-to-End Delay: Figure 6 gives the results of End-to-End delay. According to ITU-T recommendations, delay for all applications should be kept below 300ms based on this rating: [0-200]ms=Good, [200-300]ms=Acceptable, and over 300ms=poor. However, for real-time applications, delays should be kept below 150ms so that applications and users will experience transparent interactivity. Based on this, scenario 1 and 2 are absolutely unacceptable, results of scenario 3 and 4 are very satisfying (largely below 150ms). But comparing to the delays in P2P communications, scenario 4 provides the shortest delays comparable to P2P communications.

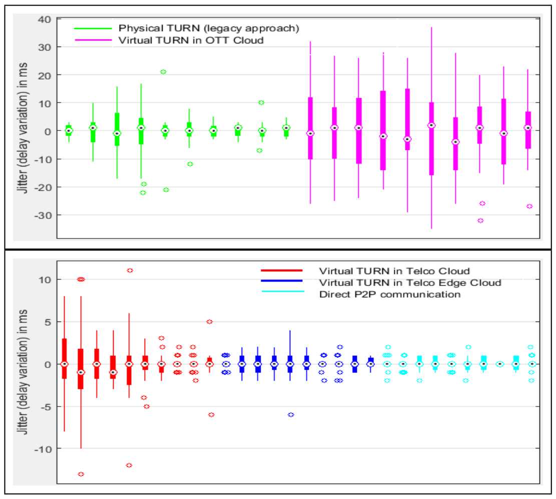

Fig. 7: Evaluation of Jitter Distribution

3) Jitter: Figure 7 gives the results of Jitter evaluation. Reliable jitter should be below 10% of average TTL and the upper limit is 10% of total RTT. For real-time applications, 95% of jitter measurements should be below 30ms and jitter=[0-10]ms=Perfect. Furthermore, as jitter is critical, jitters of less than 1ms are needed. So the more jitter is close to 0, the better is the call QoS. According to these guidelines, results of scenario 1 and 2 are overally unacceptable (above 10ms). Results of scenario 3 are very satisfying with low mean values included between [-10ms, +10ms] but they are so scattered comparing to results of scenario 4 (using the proposed approach) which are very satisfying and stable with all jitter mean values around 0 within [-5ms, +5ms], closest to P2P results where all jitter mean values are equal to 0.

D. Discussions

Our explanations for these results are that, from one side, physical network functions deployed in the WAN (scenario 1), and VNFs deployed in OTT Cloud (scenario 2), are located within the Internet and on large heterogeneous networks with different technologies and topologies with no location-awareness. This makes control flows and data flows traverse, in a best-effort mode, intermediate usually congested networks depending on the distance between end-users and servers which hugely impacts delay and jitter.

From another side, when VNFs are deployed in Telco Clouds (scenario 3) results are acceptable as these VNFs may benefit from the networking knowledge Telcos have for better cloud networking and thus, somehow, flows are managed (not in best-effort). So, from scenario 1 to 3, we conclude that the existing approaches are not sufficient: real-time interactions are compromised of QoS. At a further side, beyond the obvious reduction of time to deploy, to dimension and to configure physical network functions comparing to deploying a VNF; the time to deploy a VNF is reduced when using containers instead of VMs and this has been proven by our results on call setup time measurements.

Furthermore, VNFs deployed at the Telco Edge Cloud using location-awareness benefit from reduced users-to-Cloud distances as VNFs are located at the edge Telco networks close to users. Thus, control and data flows traverse a minimum number of hops which explains the very satisfying low call setup time, delay and jitter results (almost P2P).

VI. Conclusions & Future Work

We have presented a Deployment Model with an NFV-based exposition for an on-demand deployment of VNFs in the context of operator openness towards OTTs. The model proposes to rely on dynamicity offered by the convergence of control and management planes. Network APIs are exposed through Management and Orchestration modules and VNFs are deployed on-demand at the Telco Edge. We have shown the achieved dynamicity in applications-to-network demand and in VNF deployment at the network Edge PoPs.

As a next step, we study flexible VNF deployment and re-deployment according to different user mobility constraints.

References

[1] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, “Network function virtualization: Challenges and opportunities for innovations,” *Communications Magazine, IEEE*, vol. 53, pp. 90–97, Feb 2015.
[2] A. Boubendir, E. Bertin, and N. Simoni, “Naas architecture through sdn-enabled nfv - network openness towards communication service providers,” in *NOMS 2016*, (Istambul, Turkey), April 2016.
[3] *ETSI NFV ISG, Network Functions Virtualisation (NFV); Management and Orchestration. ETSI GS NFV-MAN 001 V1.1.1. http://www.etsi.org/technologies-clusters/technologies/nfv*, Dec 2014.
[4] W. Shi and S. Dustdar, “The promise of edge computing,” *Computer*, vol. 49, pp. 78–81, May 2016.
[5] G. Xilouris, et al., “T-nova: A marketplace for virtualized network functions,” in *Networks and Communications (EuCNC), 2014 European Conference on*, pp. 1–5, June 2014.
[6] M. J. McGrath, et al., “Performant deployment of a virtualised network functions in a data center environment using resource aware scheduling,” in *IFIP/IEEE IM Symposium*, pp. 1131–1132, May 2015.
[7] G. Xilouris, et al., “T-nova: Network functions as-a-service over virtualised infrastructures,” in *IEEE NFV-SDN Conference*, Nov 2015.
[8] ONF, *Solution Brief, OpenFlow-Enabled SDN and Network Functions Virtualization*, February 2014.
[9] ONF, *Solution Brief, Operator Network Monetization Through OpenFlow-Enabled*, April 2013.
[10] “Open mobile alliance. http://technical.openmobilealliance.org/technical/.”
[11] K. Liu and K. Xu, “Open service-aware mobile network api for 3rd party control of network qos,” in *Computer Science and Electronics Engineering (ICCSEE), International Conference on*, vol. 1, 2012.
[12] R. Ravindran, X. Liu, A. Chakraborti, X. Zhang, and G. Wang, “Towards software defined icn based edge-cloud services,” in *Cloud Networking (CloudNet), IEEE 2nd International Conference on*, pp. 227–235, 2013.
[13] M. Lanthaler and C. Gutl, “Toward a restful service ecosystem,” in *IEEE Conference on Digital Ecosystems and Technologies*, pp. 209–214, 2010.
[14] C. Anderson, “Docker [software engineering],” *IEEE Software*, vol. 32, pp. 102–c3, May 2015.
[15] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene, “An open framework to enable netfate (network functions at the edge),” in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pp. 1–6, April 2015.
[16] B. I. Ismail and al., “Evaluation of docker as edge computing platform,” in *Open Systems (ICOS), IEEE Confernece on*, pp. 130–135, Aug 2015.
[17] *OPNFV, Linux Foundation Collaborative Project. PAVING THE WAY TO OPEN SOURCE NFV*, June 2015.
[18] “Open mobile alliance. guidelines for restful network apis. http://technical.openmobilealliance.org/technical/,” February 2014.
[19] C. Jennings and al., “Real-time communications for the web,” *Communications Magazine, IEEE*, vol. 51, pp. 20–26, April 2013.
[20] callstats.io, “[online] available on: http://www.callstats.io/.”