ForCES Applicability to SDN-enhanced NFV

Evangelos Haleplidis, Spyros Denazis, Odysseas Koufopavlou
Electrical & Computer Engineering Department
University of Patras
Rio, Greece
[ehalep, odysseas]@ece.upatras.gr, sdena@upatras.gr

Damascene Joachimpillai
Verizon
Waltham, MA
USA
dj@verizon.com

Jamal Hadi Salim
Mojatatu Networks
Ottawa, ON
Canada
hadi@mojatatu.com

Diego Lopez
Telefónica I+D
Madrid
Spain
diego@tid.es

Jason Martin
Cumulus Networks
Mountain View, CA
USA
Jason@cumulusnetworks.com

Kostas Pentikousis
EICT
Berlin
Germany
k.pentikousis@eict.de

Abstract— **Networking has seen lately a surge in research and innovation with the re-emergence of network programmability in the form of Software-Defined Networking (SDN), a new approach for network datapath configuration. SDN provides an abstraction model of the Forwarding Plane and separates it from the Control Plane using open APIs. In parallel, major telecom operators have embarked on an effort to bring the advantages of virtualization to carrier network infrastructures. Part of this effort was invested in establishing the Network Function Virtualization (NFV) Industry Specification Group (ISG) at the European Telecommunications Standards Institute (ETSI). The NFV goal is to define how Network Functions (ranging from firewalls and load-balancers to routers and access elements) can be virtualized and run as software on high-volume servers instead of specialized hardware. This paper treats SDN and NFV as complementary concepts that together form a bigger picture in the domain of future carrier networks and discusses the complete lifecycle of such a network. In this context we present how ForCES can be used as the foundation for SDN-enhanced NFV and describe the blueprint for the Proof of Concept (PoC) prototype which has been introduced to the NFV ISG. A key goal of this paper is to concisely position carrier NFV and SDN activities under a unified framework.**

Keywords-component; Software Defined Networking, Network Function Virtualization, Abstraction Model, IETF, ETSI, ForCES, PoC

I. Introduction

With the advent of cloud computing and virtualized infrastructure in the data centers, elasticity amongst others is important for both operational and capital expenses for operators and service providers. Motivated by the advances of virtualization, since October 2012, several major telecom operators drafted the first white paper for Network Functions Virtualization (NFV) [1]. In short, the problem that NFV addresses is three-fold. First, NFV aims to reduce the costs of purchasing physical, dedicated, and expensive network equipment such as firewalls, which may be sparsely used as the purchase is based on perceived use and, in practice they remain underused, increasing the overall energy consumption in infrastructure networks. Second, NFV should enable shorter innovation and deployment cycles for new network functionality that can provide new user services. Third, NFV aims to reduce the overall management cost required for a large and dynamic number of such devices through automation.

NFV's solution is to virtualize network functions and deploy them as software, e.g. as virtual machines (VM) and functions, running on high-volume devices on an as-needed basis. As stated in [1], NFV is applicable to any data plane packet processing and control plane function in mobile and fixed networks. Such an approach will enable operators to design, implement, deploy and destroy network functions at will and reduce energy consumption by utilizing only the necessary amount of infrastructure resources similar to [2]. In order to do so, NFV must provide both flexibility and adaptability for new user requirements in service delivery. In short, a main NFV tenet is the separation of functionality (in software) and capacity (in virtualized hardware).

On the other hand of the networking spectrum, research and innovation in network programmability have been reignited in the form of SDN, see [3] and references therein. SDN, in a few words, refers to the ability of software applications to program individual network devices and, in effect, the network as a whole via open, standardized interfaces. One of the key elements in SDN is the separation of the control from the forwarding plane by defining a common abstraction model of the forwarding plane accompanied by one or more protocols to perform the control and configuration. Such an abstraction model decouples both planes and allows faster innovation as well as interoperability between vendors. In SDN, a controller provides access to the resources of the forwarding plane to applications that reside on top.

SDN and NFV are complementary: The former decouples control from the datapath network functions and the later decouples placement of the network functions and their control from the underlying hardware. SDN owns the programmatic control and configuration of the devices and network functions while NFV deals with the lifecycle of those Network Functions (NFs). As an example of the interaction between SDN and NFV, the instantiation of new Network Functions falls in the domain of NFV, while the integration within the network and the automation of forwarding traffic to newly instantiated Network Functions fall into the realm of SDN. While NFV can work with traditional methods of network management, SDN will provide the necessary tools for rapid deployment of NFV.

Cumulatively, seeing the need for an abstract reference model for both SDN and NFV and anticipating, as stated, a tight correlation between SDN and NFV we argue that there is a need for a unified network abstraction model that will encompass both SDN and NFV. Such a model will become more ubiquitous if we consider arbitrary packet processing functions, such as routers and switches, which can be virtualized as Network Functions and become part of a virtualized network infrastructure. Earlier work has presented the main rationale in [4], [5]. Such a model will enable us to have a single point of reference for the network elements' lifecycle. The model ought to be open, extensible and flexible enough in order to describe these elements from both SDN and NFV. In addition the model must allow easier integration of new forwarding capabilities and network functions and allow tools to span both SDN and NFV allowing the creation of applications such as a global network manager. A key contribution of this paper is to document the recently submitted Proof-of-Concept (Poc) [6] to the NFV and the realization of how the common model and architecture can converge the SDN and NFV domains. The PoC employs the IETF Forwarding and Control Element Separation (ForCES) framework [7]. While multiple PoCs have been submitted to the NFV ISG (e.g. [8] and [9]), to the best of our knowledge, we have not seen a similar PoC that opens up NFs using SDN techniques or addresses the NFV architecture using a single framework.

This paper makes the following three contributions. First, we provide feedback to ongoing standardization efforts at both the IETF and NFV ISG communities. For the IETF ForCES working group this paper provides additional implementation experience and possibly new requirements that can be useful in a future re-chartering process. For the NFV ISG, this work will provide valuable feedback to the working groups focused on architecture specification and Management and Orchestration (MANO) issues. Second, this paper evaluates the implementation feasibility of SDN on the NFV architecture whilst using a single framework, which simplifies design and implementation cycles needed to develop Network Functions and NFV altogether. Finally this work showcases the applicability of ForCES in a different context from the one that it was initially designed for. As we discuss later on, ForCES was initially aimed for separating the forwarding and control planes, but the extendable model makes ForCES a versatile framework.

This remainder of this paper is structured as follows. The following section serves as a ForCES primer, followed by a brief introduction into the NFV published architecture. We continue by presenting the mapping of ForCES to the NFV architecture and conclude with expectations and future work.

II. A ForCES Primer

The IETF ForCES working group was initially chartered with the goal to separate the forwarding from the control plane. The ForCES framework [7], as seen in Fig. 1, defines two main elements, namely, Control Elements (CEs) and Forwarding Elements (FEs). In this framework, CEs control the FEs. Additionally the framework defines two helper elements, the CE and FE managers assist in the bootstrapping phase. The CE manager is currently out of scope of the working group, and the FE manager has recently been added to the charter.

ForCES defines an object oriented model [10], also referred to as a modeling language, realized by an XML schema. The goal of the ForCES model is to abstract FE resources. The modeling language is used to construct XML models, also referred to as libraries, of resources of the datapath up to a very fine detail and is based on a building block approach similar to [11]. Each building block is an object class known as a *Logical Functional Block* (LFB). When acting on packet processing resources, LFB class instances receive, process, modify, and transmit packets. These LFB class instances can be interconnected in a directed graph to form a service. An LFB class instance can be thought of as a block that performs a well-defined action or computation on the packets passing through it such as, for example, classifiers, shapers and meters.

Each LFB class defines input and output ports, operational parameters visible to a CE, capabilities advertised to the CE, and events that a CE can subscribe to. Components and capabilities are modeled using data types. These data types can be atomic, or can be compound, such as structs and arrays. LFB classes can be inherited and extended or overridden thus allowing definition reuse. Versioning allows for both forward and backward compatibility.

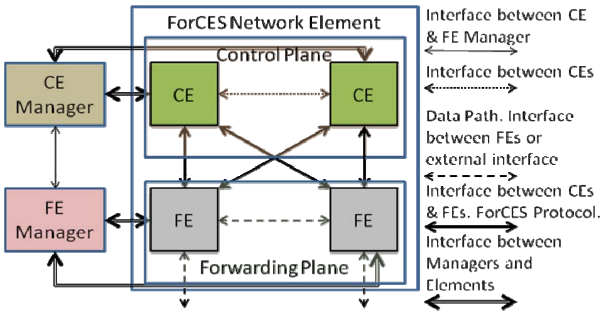
ForCES Network Element
Control Plane
CE Manager
CE
CE
FE Manager
FE
FE
Forwarding Plane
Interface between CE & FE Manager
Interface between CEs
Data Path. Interface between FEs or external interface
Interface between CEs & FEs. ForCES Protocol.
Interface between Managers and Elements

Figure 1. ForCES framework

The modeling language defines base data types, similar to the C programming language, e.g. ints, chars and strings; using these base data types a developer can define new custom data types to suit the needs of any problem. While the initial concept of an LFB Class definition was to be a resource representation block for fine-grained operation of the forwarding plane by the control plane, it has found uses in more general contexts because the definition the ForCES model is sufficiently flexible. LFBs can be used to define various configurations and can be as fine- or coarse-grained as needed.

One of the major advantages is that the ForCES model allows the definition of new LFBs, each with its own customized set of parameters. As an example of the model's expressiveness, the graph of the LFBs has already been modeled and can be manipulated, effectively changing the functionality of the FE. An additional example includes the modeling of the LFB class topologies as tied to specific FE implementations allowing a CE to discover how LFBs can be stitched together in a viable graph. The ForCES data model is complemented with a protocol [12]. The protocol by design has very few and simple commands which act on the underlying specified LFB class models. The main advantage of the protocol is that it is model-agnostic and, thus, can control and configure any FE that is modeled with the ForCES model without the need for changes.

The ForCES protocol comes with a rich functionality necessary for providing robust and efficient control of the underlying resources: these include high availability, controllable heartbeat mechanism, transactions including two phase commits, various execution modes and command batching and pipelining. In addition, the ForCES protocol comes with a concise yet complete set of "verbs": SET, GET and DELETE, REPORT and REDIRECT. Combined with the unlimited amount of data defined by the LFB class developers, ForCES emerges as a language with easy-to-understand semantics.

Addressing the LFB components, capabilities and events, with the protocol, follows a tree-like hierarchy with unique 32-bit identifiers for each step, similar to SNMP's OID notation, beginning at the start of the LFB definition. As an example, consider the following LFB definition which illustrates the different entities of an LFB class.

```
<LFBClassDef LFBClassID="100">
  <name>SimpleLFB</name>
  <synopsis>A simple LFB</synopsis>
  <version>1.0</version>
  <components>
    <component componentID="1" access="read-only">
      <name>GoodPackets</name>
      <synopsis>A packet counter</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="2" access="read-only">
      <name>BadPackets</name>
      <synopsis>Bad packet counter</synopsis>
      <typeRef>uint32</typeRef>
    </component>
  </components>
  <capabilities>
    <capability componentID="3">
      <name>CheckingTypes</name>
      <synopsis>Bad packet types</synopsis>
      <array>
        <atomic>
          <baseType>uchar</baseType>
          <specialValues>
            <specialValue value="1">
              <name>Checksum</name>
              <synopsis>IPv4 checksum</synopsis>
            </specialValue>
            <specialValue value="2">
              <name>BadFrame</name>
              <synopsis>Bad MAC Frame</synopsis>
            </specialValue>
          </specialValues>
        </atomic>
      </array>
    </capability>
  </capabilities>
  <events baseID="4">
    <event eventID="1">
      <name>TooManyBadPackets</name>
      <synopsis>Too many Bad packets</synopsis>
      <eventTarget>
        <eventField>BadPackets</eventField>
      </eventTarget>
      <eventGreaterThan>100</eventGreaterThan>
      <eventReports>
        <eventReport>
          <eventField>BadPackets</eventField>
        </eventReport>
      </eventReports>
    </event>
  </events>
</LFBClassDef>
```

This example LFB has two components. Each component is a counter, one for "good" and one for "bad" packets, each an unsigned 32-bit integer and are read-only, meaning that a CE cannot set any value. This LFB also defines a capability that will let the CE know which kind of checks it is able to perform on the packets. Specifically it defines the capability to check IPv4 checksum and MAC frames. Finally it has defined an event where the CE will be notified when the bad packets counter exceeds a specific value, in this case initially 100. This value is configured by the ForCES protocol.

LFB models are classes and their instances exist (are instantiated) in an FE. For example, if the CE would like to retrieve the number of the good packets counter from a specific FE, it would issue a GET command and specify the FEID, e.g. 10, followed by the LFBClass, in this case 100, the LFB instance, e.g. 1, followed by the component ID. Thus a tuple of {10,100,1,1} would form a unique NE cluster-wide path to the component. Since both counters are read-only, a SET or DELETE command to the same path will return an error. In addition, the CE may request the capabilities, by simply using the command GET to the path tuple {10,100,1,3}. If the CE wanted a specific row of the table it would suffix the row index into the path thus making it a GET to the path tuple {10,100,1,3,1}. The last command would retrieve the first row of the capabilities table.

In essence for the protocol and the CEs the LFB XML models are an abstraction of the forwarding plane and become the point of interoperability between different implementations. It is worth noting that using the same LFB model for different hardware architectures, e.g. IA x86, x64, ARM, allows the use of the same control elements and will treat the LFB the same. Similarly, the LFB model does not make any distinction of whether LFBs are physical or virtual and does not have any impact on the design of the CEs thus expanding the range of what can be considered an LFB.

III. Network Functions Virtualization

As discussed earlier, driven by the advances of virtualization and with the goal of reducing CAPEX and

OPEX, some of the world's major operators got together and drafted the white paper for NFV [1]. NFV's goal was not to standardize but rather to provide an architecture framework based on the production of a number of whitepapers and requirements documents, describing the problem scope and outlining the solutions to achieve the aforementioned goals.

The NFV has already published the first batch of its specifications, addressing amongst others, the terminology [13], as well as the NFV architecture framework [14]. The architecture as defined by the NFV ISG is seen in Fig 2.

The architecture elements include the virtualization layer that hides the computing, storage and network hardware elements and provide a uniform approach to the virtualized infrastructure manager to instantiate the respective virtual resources, namely computing, storage and network. Usually these resources are provided through a hypervisor.

The ISG architecture also contains the Virtual Network Functions (VNF) which is a virtualization of a regular Network Function (NF). Examples of NFs include firewalls, switches, routers but it can also contain more broad range of functions, such as the 3GPP's PGW and SGW which we will elaborate further. A VNF can also be managed and controlled by an Element Management System (EMS). An EMS may control one or more VNFs. EMSs appear in the NFV architecture to show the requirement for VNF configuration and management in an orthogonal way to the pure virtualization aspects. In our view we see EMSs as one or more VNFs and will refer to them further as Element Management Functions (EMFs).

Finally the architecture includes the Management and Orchestration (MANO) of the NFs and the NFV infrastructure. The MANO is comprised of three components, the Infrastructure managers responsible for managing the infrastructure resources, the VNF managers responsible for managing the VNFs lifecycle and the Orchestrator which is in charge of the orchestration and management of FNV infrastructure and network services.

The unnamed interfaces between elements in the architecture definition of the NFV ISG are declared out of scope of the specification. Thus one of the main contributions of this paper and PoC is the inclusion of the interface between the VNF and EMS as a standard interface, providing a similar to SDN interface for VNFs.

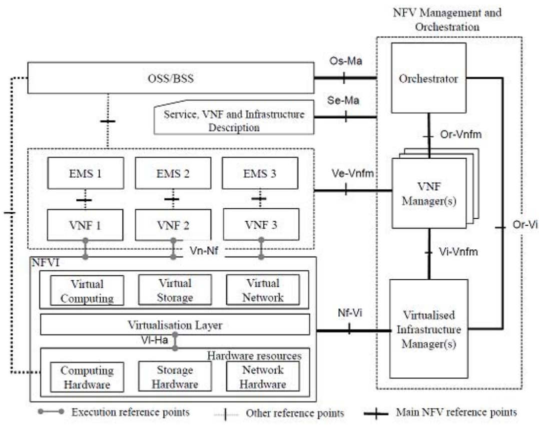

Figure 2. NFV ISG architecture

IV. FoRCES Applicability Proof of Concept

As discussed in the introduction, conceptually and technically, SDN and NFV can be knitted together to provide a bigger picture of the network, addressing a complete network elements' lifecycle. NFV can then provide a flexible infrastructure substrate by deploying new NFs, while SDN can be responsible for configuring the infrastructure's datapath. Additionally, as discussed earlier, packet processing appliances can be realized as NFs, empowering NFV to be able to create virtual network infrastructures and SDN to configure them, thus having one complete virtual network manager. Fig. 3 depicts how ForCES maps on the NFV ISG architecture in our PoC [6].

For the networking interconnection of the various instantiated network functions, denoted with the red rectangles, we will implement a series of LFBs. Example of networking LFBs includes, a bridge LFB for interconnecting VNFs within the networking infrastructure, a port LFB to specify the connectivity of the VNFs in the virtual network and a tunnel LFB, specifically the GTP tunneling for the implemented VNFs as we will elaborate further.

When a user attempts to use a network function, a control or management application will be able to authorize the user's flows, if the user is allowed to use that specific network function. This path provisioning can be executed either on a hop-by-hop basis or even in "one go" across an authorized path. Such an approach will provides for complete isolation and protection across different network functions and naturally supports multi-tenancy as the decisions are made from a central point with complete visibility to the system.

For the compute and storage facilities we will define a hypervisor LFB that will receive ForCES commands and instantiate virtual environments. The hypervisor LFB is able to support instantiation of both virtual machines, using Linux libvirt, as well as linux namespaces. A first prototype implementation was demonstrated at IETF 84 [15]. By simply augmenting the hypervisor LFB, we plan to support more virtualization frameworks.

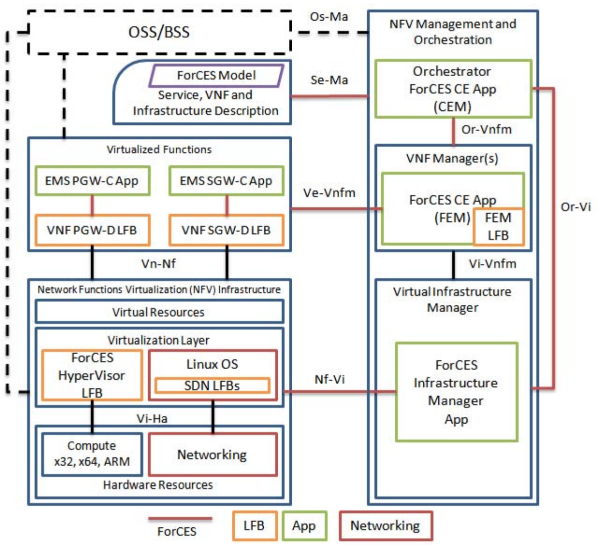

Figure 3. ForCES mapping on the NFV ISG architecture

Thus, both the hypervisor and packet processing LFBs will be controlled and managed by the ForCES infrastructure manager application. The aforementioned application will be responsible for instantiating virtual environments and handling their interconnections as requested by the orchestrator and VNF managers. The infrastructure manager is an application on top of a CE.

Our approach is very similar to other infrastructure provisioning frameworks, e.g. OpenStack and CloudStack. As discussed also in [5], ForCES provides a uniform approach for NFV and SDN instead of having to use multiple protocols and frameworks to achieve the same result. ForCES provides a common framework and one consistent API for all the tasks required for NFV and SDN. In addition ForCES natural extensibility and expresibility over that of OpenFlow [16] makes ForCES a viable candidate for separating the control from forwarding for Network Functions, a concept we also explore in this paper. An interesting point for further research would be to put ForCES on top and/or below said infrastructure frameworks, given the recursive nature of this virtualization approach.

In regards to specific network functions, we aim to demonstrate at EWSDN 2014 the instantiation of prototypes of the Packet Gateway (PGW) and the Service Gateway (SGW) of the Third Generation Partnership Project's (3GPP) Evolved Packet Core (EPC) [17]. An early demonstration of this work was done at IETF 88, gathering significant interest from all quarters of the IETF mobility management working groups. In essence, SGW routes and forwards user data packets, while also acting as the mobility anchor for the user plane during inter-eNB handovers. PGW enforces quality-of-service (QoS) policies and monitors traffic to perform billing to various policy enforcement functions of the EPC. The PGW also filters packets and connects to the Packet Data Network (PDN), i.e. network services as well as access to the Internet and other cellular data networks and PDNs, and includes services like firewalls and deep packet inspection. Data traffic between the SGW and PGW is sent via the GTP-U protocol while signaling traffic between SGW and PGW is sent via the GTP-C protocol.

In the PoC prototype, we have split the data and signaling part of the PGW and SGW into the PGW-D and SGW-D and PGW-C and SGW-C, respectively, an approach followed also in [18]. S/PGW-D handle the tunnel encapsulation and decapsulation as well as the analytics collection while S/PGW-C are responsible for setting up the tunnel endpoints, handling the signaling between these entities in conformance with the 3GPP specification for GTP-C. S/PGW-C are also responsible for communicating with all the relevant to the LTE architecture policy mechanisms for maintaining subscriber policies. For instance the following is the XML datatype definition of a tunnel endpoint specifically for our PoC.

```
<dataTypeDef>
   <name>gtpTableEntry</name>
   <synopsis>Table Entry for the GTPv1-U LFB</synopsis>
   <struct>
      <component componentID="1">
         <name>UEIP</name>
         <synopsis>IP Address of the User
Equipment</synopsis>
         <typeRef>IPv4Address</typeRef>
      </component>
      <component componentID="2">
         <name>SourceIP</name>
         <synopsis>IP address of the sender</synopsis>
         <typeRef>IPv4Address</typeRef>
      </component>
      <component componentID="3">
         <name>DestinationIP</name>
         <synopsis>IP address of the
destination</synopsis>
         <typeRef>IPv4Address</typeRef>
      </component>
      <component componentID="4">
         <name>TEIDSource</name>
         <synopsis>TEID source</synopsis>
         <typeRef>uint32</typeRef>
      </component>
      <component componentID="5">
         <name>TEIDDestination</name>
         <synopsis>TEID destination</synopsis>
         <typeRef>uint32</typeRef>
      </component>
   </struct>
</dataTypeDef>
```

This definition describes a structure comprised of five components. The IP address of a user's equipment with which we identify traffic from the user, the IP address of the source tunnel endpoint and the destination IP address of the remote tunnel endpoint as well as the tunnel endpoints, source and destination, IDs.

Using the ForCES model, we can apply SDN capabilities to the network functions as well. In our view ForCES LFBs, the data part of NFV, are NFs, and ForCES CEs and CE applications as EMFs that can converge both the infrastructural network management (at the bottom layer) and VNF functionality management through a common framework.

Our approach can accrue the same benefits of separation in terms of splitting the control and data part and relocating them as needed. This approach opens up possibilities for scaling up and down the signaling and data parts of the functions as needed based on load or even energy consumption in an elastic manner. Finally this approach opens the market for developers who can specialize on specific functionalities while retaining interoperability with the respective partners.

In addition, using our approach enables service providers with the ability to select and instantiate LFBs dynamically as needed using the concepts of service chaining [19]. For example the SGi-LAN, which includes all the network services included in the PDN where normally service providers would like to create dynamically chains of services. They can select LFBs instantiate them as LFBs and interconnect them and control and manage them in a uniform approach.

Network Functions are instantiated by a ForCES CE application, the FE Manager (FEM), which is responsible for selecting where the VNFs will be instantiated and request to the ForCES infrastructure manager application to instantiate virtual environments. FEM will then instruct the instantiated NFs with the necessary bootstraping information, including which NFs to be connected to in order to function properly. Standardization work on the FEM is already under way in the ForCES working group and we intend to provide valuable feedback based on our ongoing implementation work.

The orchestrator of the whole system, a ForCES application, will provide the ability to select the desired NFs. The NFs, the service and the infrastructure description will be modeled using the ForCES model.

Already an LFB exists that contains the graph of LFBs which can be in turn manipulated by a CE. Extrapolating from that approach we can model graphs of FEs, or even graphs of NFs as well as graphs of interconnected virtual environments to form the infrastructure's description.

An example of the PoC's workflow starts with the request from the orchestrator app to instantiate the SGW and PGW applications and datapath functionalities (VNFs). The Infrastructure manager will instantiate virtual containers, setup their interfaces and interconnect them followed by the instantiation of the VNFs by the VNF manager application. Once the VNFs start, the applications will control the datapath and perform normal operation.

V. Conclusions and Future Work

In this paper we presented the applicability of ForCES upon the NFV architecture and how the ForCES framework can provide a common approach on NFV as well as enhance with SDN concepts. We also described our Proof-of-concept submission to the NFV ISG. We elaborated on our three potential contributions to the IETF and NFV standardization activities namely the feedback to ongoing standardization efforts, evaluation of the implementation feasibility of SDN on the NFV architecture whilst using a single framework and showcase the applicability of ForCES in a different context from the one that it was initially designed for. The initial work that led to the submission of the PoC to the NFV ISG was demonstrated at the IETF 88 in Vancouver. This PoC has been demonstrated at the IETF 90 in Toronto at the end of July 2014. We aim to demonstrate a further improved prototype at EWSDN 2014.

The choice of using ForCES instead of any other combination of frameworks, was driven by the extensibility and the expressiveness of the ForCES model. Using ForCES we're able to address NFV and SDN issues in their entirety, instead of focusing on one problem at a time such as in [20]. Having one model and a consistent set of APIs, instead of using multiple APIs, simplifies implementation, eases the learning curve and reduces interdependencies, e.g. the need to develop plugins to interconnect different frameworks.

This prototype PoC will provide initial results of the applicability of ForCES on NFV as well as enhancing NFV with SDN techniques. Future work is needed to complete the implementation for a fully working NFV infrastructure and for further implementation of other NFs. Support for more virtualization environments using the same/common hypervisor LFB is also in our work agenda.

References

[1] http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[2] Martins, Joao, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. "ClickOS and the art of network function virtualization." In Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)}, pp. 459-473. USENIX}, 2014.

[3] Evangelos Haleplidis, Spyros Denazis, Kostas Pentikousis, Jamal Hadi Salim, David Meyer and Odysseas Koufopavlou, "SDN Layers and Architectures Terminology", http://tools.ietf.org/html/draft-haleplidis-sdnrg-layer-terminology-07, Individual submission for IRTF SDNRG research group, July 2014, work in progress.

[4] ETSI Presentation on SDN and NFV, April 2013. http://docbox.etsi.org/Workshop/2013/201304_FNTWORKSHOP/S06_SNDpart2/UNIofPATRAS_HALEPLIDIS.pdf

[5] Evangelos Haleplidis, Jamal Hadi Salim, Spyros Denazis, and Odysseas Koufopavlou., "Towards a Network Abstraction Model for SDN.", Journal of Network and Systems Management (2014): 1-19. Special Issue on Management of Software Defined Networks, Springer, 2014, doi:10.1007/s10922-014-9319-3

[6] Jamal Hadi Salim, Damascene Joachimpillai, Jason Martin, Diego Lopez, Haleplidis Evangelos, "ForCES applicability for NFV and integrated SDN", ETSI NFV PoC, April 2014, http://docbox.etsi.org/ISG/NFV/PER/05-CONTRIBUTIONS/2014//NFVPER(14)000046r2_ForCES_Applicability_for_NFV_and_integrated_SDN.docx

[7] Yang, L., Dantu, R., Anderson, T., & Gopal, R. (2004). Forwarding and control element separation (ForCES) framework. RFC3746.

[8] CloudNFV Open NFV Framework "CloudNFV PoC proposal", http://docbox.etsi.org/ISG/NFV/PER/05-CONTRIBUTIONS/2013/NFVPER(13)000040r1_CloudNFV_PoC_Application.docx

[9] Kevin McBride et al., A.1 NFV ISG PoC Proposal, "Multi-vendor Distributed NFV", http://nfvwiki.etsi.org/images/NFVPER(14)000011_NFV_ISG_PoC_Proposal_-_Multi-vendor_Distributed_NFV.pdf

[10] Joel Halpern and Jamal Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.

[11] Kohler, E., Morris, R., Chen, B., Jannotti, J., & Kaashoek, M. F. (2000). The Click modular router. ACM Transactions on Computer Systems (TOCS), 18(3), 263-297.

[12] Avri Doria, Jamal Hadi Salim, Robert Haas, Horzmud Khosravi, Weiming Wang, Ligang Dong, Ram Gopal and Joel Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, March 2010.

[13] European Telecommunication Standards Institute, "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV", white paper, ETSI GS NFV 003, 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.01.01_60/gs_NFV003v010101p.pdf

[14] European Telecommunication Standards Institute, "Network Functions Virtualisation (NFV); Architectural Framework", white paper, ETSI GS NFV 002, 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.01.01_60/gs_NFV003v010101p.pdf

[15] Jamal Hadi Salim "FEM presentation in IETF 84". http://www.ietf.org/proceedings/84/slides/slides-84-forces-2.pdf

[16] Evangelos Haleplidis, Spyros Denazis, Odysseas Koufopavlou, Joel Halpern, and Jamal Hadi Salim. "Software-Defined Networking: Experimenting with the control to forwarding plane interface." In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pp. 91-96. IEEE, 2012.

[17] Olsson, Magnus, Stefan Rommer, Catherine Mulligan, Shabnam Sultana, and Lars Frid. "SAE and the Evolved Packet Core: Driving the mobile broadband revolution". Access Online via Elsevier, 2009.

[18] K. Pentikousis, Y. Wang, and W. Hu, "MobileFlow: Toward software-defined mobile networks." Communications Magazine, IEEE 51, no. 7 (2013).

[19] W. John, et al., "Research Directions in Network Service Chaining", Proc. IEEE SDN4FNS, November 2013.

[20] Gember-Jacobson, Aaron, Chaithan Prakash Raajay Viswanathan, Robert Grandl, Junaid Khalid, Sourav Das, and Aditya Akella. "OpenNF: Enabling Innovation in Network Function Control." ACM SIGCOMM August 2014.