Leveraging NFV for the Deployment of NDN: Application to HTTP Traffic Transport

Xavier Marchal[1], Moustapha El Aoun[2], Bertrand Mathieu[3], Thibault Cholez[1], Guillaume Doyen[2], Wissam Mallouli[4], Olivier Festor[1]

[1]University of Lorraine, LORIA, UMR 7503 Nancy, France

[2]Troyes University of Technology, ICD, UMR CNRS 6281 Troyes, France

[3]Orange Labs Lannion, France

[4]Montimage Paris, France

***Abstract*—For a few years, Network-Function Virtualization (NFV) acts as the most promising solution for the flexible implementation and management of future network services. If most of current efforts in this area focus on IP-based Virtual Network Functions (VNF), the case of Information-Centric Networking (ICN) is interesting since it can demonstrate that NFV is a promising technology for ISP to deploy such new innovative network stacks. In this context, we propose to design and implement a NFV compliant architecture to easily deploy ICN islands. Especially, at the core of this architecture, we present an HTTP/NDN gateway, which enables our network to carry real HTTP traffic. Finally, we show early functional experimental results of an initial testbed deployment exhibiting the capability of our global infrastructure to retrieve the top-1000 of the most popular web sites.**

Keywords—Network Function Virtualization; Named Data Networking; experimental study; Web traffic

I. INTRODUCTION

Internet pursues its fast paced evolution while more and more users and bandwidth-consuming services depend on it, putting a high pressure on the underlying infrastructure. To address this challenge, several disruptive technologies have emerged from the networking community. From one side, based on the observation that the main usage of today's Internet is related to content diffusion, Information-Centric Networking (ICN) architectures, and particularly the Named-Data Networking (NDN) initiative, propose a paradigm shift to optimize data delivery, where the Internet data plane shifts from host-based network mechanisms to content-based ones and include caching on all network devices. From the other side, NFV (Network Function Virtualization), as defined by the ETSI [1], can reduce operating cost and foster the innovation in the network core by replacing specialized and expensive dedicated network equipment by commodity servers running Virtualized Network Functions (VNF).

In this context, we believe that NDN could only be deployed incrementally, step after step, at low cost for reducing network investment by network operators, and that running NDN nodes in an NFV infrastructure is the only reasonable path leading to a progressive deployment by ISPs of this new network protocol. Therefore, our research question encompasses the deployment of NDN-based VNF in an operational context. In this paper, we propose a HTTP/NDN gateway to interconnect ICN islands to the IP world, and an experimental architecture supporting this research direction and able to process the web traffic passing through a virtualized NDN network.

The rest of the paper is organized as follows. Section II presents the related work on NFV and ICN deployments. Section III describes our architecture with an emphasis on the design of the key component that is the HTTP/NDN gateway. Section IV validates our approach with a first analysis of web traffic delivered through our testbed. Finally, Section V concludes the paper and gives directions for future work.

II. RELATED WORK

A. NFV Deployment Initiatives

Currently, several efforts aim at providing implementation of the NFV reference architecture provided by the ETSI [1]. The OPNFV consortium proposes to gather all existing components, which can be integrated in a modular architecture, to produce a complete NFV implementation. They rely on both OpenStack and OpenDaylight, and, to date, their fourth version of the architecture "Danube", is able to address many NFV scenarios. Concurrently, the Telco working-group, which takes part of the OpenStack consortium, is devoted to the evolution of OpenStack toward NFV support. The methodology of the working group consists in considering the ETSI NFV use-cases as primary requirements and address, according to established priorities, the development/adaptation of elements in the OpenStack reference architecture. We also note that academics take part of these efforts with implementation proposals such as OpenANFV [2] and Cloud4NFV [3] that especially focus on automation for NFV management, and infrastructure elasticity. Only a few recent papers consider NFV to deploy ICN. In particular, Ueda et al. [14], from the ICN2020 project, focus on the performance aspects of deploying ICN routers as VNF.

B. Leveraging SDN for the deployment of ICN

Given the maturity of ICN solutions and the availability of early-implementations, several efforts are currently conducted to move ICN from a lab-restricted infrastructure to a fully deployable one, in an operated context. To that aim, SDN is often used to bring required control facilities needed to run a

non-IP stack. However, such a coupling induces a set of challenges that we briefly provide here: (1) capability to carry and control both ICN and IP traffic; (2) the compatibility with current SDN standard [4] (e.g. OpenFlow); (3) network protocols encapsulation strategy and (4) binding of the solution with a topological location (i.e. dedicated island, transit operator network, service provider, etc.). As an example of such architecture, CONET [5] proposes to deploy ICN in Openflow 1.0 transit networks. The protocol encapsulation is based on the computing of a hash value of content names inserted into UDP headers. With the same idea, in [6], the authors propose to use the IP header to carry a similar hash, thus making IP a protocol usable for both ICN and IP protocol stacks and avoiding any change in Openflow at the price of completely re-interpreting the meaning of IP header fields. By contrast, NDNFlow [7] proposes to enable the transit of both IP and ICN traffic by using a dedicated non-Openflow channel between the routers and the controller. Finally, in [8], the authors propose to use a full-overlay solution to avoid any cohabitation between IP related functions and ICN ones.

Whatever the solution proposed, the coupling of IP technologies with ICN ones requires a protocol adaptation (e.g. data-plane protocol encapsulation, dedicated use of OpenFlow as a control framework), which either takes it away from its standard use while also impacting the infrastructure already implemented. Another approach, proposed by Moiseenko et al. [9], is to work on TCP/ICN conversion through gateways. By contrast, we think the HTTP/ICN mapping is more adapted because HTTP objects share common properties with ICN content (URI-names, cachability). In addition, we leverage virtualization and especially NFV to enable the full decoupling of the infrastructure domain from the tenant one, which can host VNFs based on heterogeneous protocol stacks. As such, we consider NFV as the key-enabler for the ICN technology and we propose to design, implement and validate a dedicated framework enabling the deployment of ICN as VNF.

III. A Virtualized ICN Carrying Web Traffic

A. Our Architecture

We propose a flexible network architecture based on NFV [1] and SDN [4] principles. The full architecture is described in [11]. We hereafter focus on the necessary blocks illustrating the HTTP/NDN use-case as illustrated in Figure 1 and was demonstrated in [10]. We focus on the NFV Infrastructure (NFVI), which enables the resource virtualization and management to host VNFs deploying an ICN protocol stack, ant more precisely on the Data plane, the control plane defining the management and orchestration (MANO) aspects being left for future work. As a computing virtualization framework, we have chosen Docker, which relies on a lightweight virtualization principle. Even if Docker offers lower isolation guarantees, this choice was driven by the better performance compared to other virtualization solutions. Indeed, our initial performance evaluation [11] showed that, in a NFV context, the throughput of Openstack/KVM VNFs is half of the throughput of Docker.

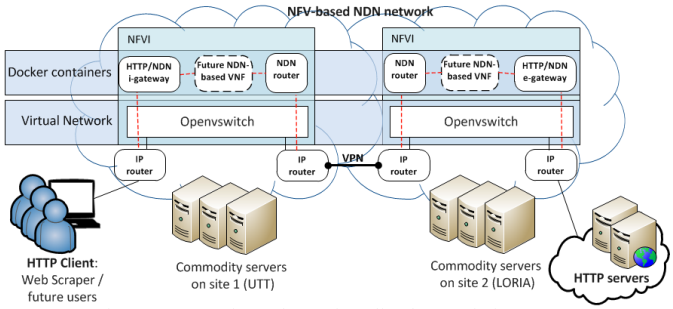

Figure 1: Overview of our virtualized NDN infrastructure

A virtual network based on OpenVSwitch is also deployed to ensure end-to-end network connectivity between the virtualized network services and enable a software control of the networking infrastructure. Then, as an ICN protocol stack, we have selected NDN [12] due to the maturity of its specification and implementation. We "dockerized" the NDNx software to make it a VNF, deployable in our architecture.

Please note that we use NDN over IP, and not directly over Ethernet because most of NFV tools are still IP-dependent. Finally, we consider the Web as our main application-layer service due to its high-popularity and predominance in the global network shares. Moreover, the benefits of transporting web traffic through an ICN have been investigated at the origin of NDN, which especially focuses on the gain that native multicasting of communications and caching can bring to massive content diffusion. However, since current web clients and servers do not yet implement NDN, and because telco operators are still reluctant to widely deploy it, we have designed and implemented dedicated gateways to perform an HTTP/NDN conversion. Those gateways are conceived as VNFs and can be deployable where and when required, thanks to NFV principles. Two kinds of gateways are defined: (1) an ingress gateway (iGW), aiming at converting HTTP users' requests into NDN *Interest* messages to find the content in the NDN network, and converting NDN *Data* messages into HTTP replies sent to the end-users; and (2) an egress gateway (eGW), the counterpart of the first one, aiming at converting NDN messages into HTTP requests towards IP web sites if the content is not available in the ICN network, and converting HTTP replies into NDN *Data* messages to the iGW.

B. HTTP to NDN translation

We now detail the two gateways we have developed, being the main VNFs running in our NFV-based testbed. First, we had to design a translation scheme to transport over NDN the main HTTP protocol header information, as described below:

1) Naming scheme. In order to communicate with each other, the gateways follow a naming pattern based on the official naming proposition to convert URL to ICN names[1].

Table 1: Naming pattern of the protocol

a	*/http/reverse_splitted_domain_name/URI/sender_route/sha1*
b	*/sender_route/sha1(/segment)*
c	*/http/reverse_splitted_domain_name/URI/sha1(/version/segment)*

[1] http://www.icn-names.net/

Table 2: Example of HTTP request translation

a	*/http/com/firefox/detectportal/success.txt /%07%0B%08%04http%08%03iGW/1E69...*
b	*/http/iGW/1E69...(/segment)*
c	*/http/com/firefox/detectportal/success.txt/1E69...(/version/segment)*

Splitting the domain per sub-level and appending them in reverse order gives us a better NDN routing capability with route aggregation compared to a "monolithic" domain name like we saw in another NDN gateway [11]. Also, by adding to the name the prefix "/http", an egress gateway can register only this prefix in order to be the default producer for all the traffic or, for example, "/http/com" prefix to be the default producer for all .com domains. Furthermore, beginning an ICN name by the protocol name "http" can later enable per protocol routing and traffic management with different strategies or routes applied to each of them.

2) Request a web content in the NDN network. Since NDN Interest packet can't carry data while HTTP request's header does, the Ingress gateway (or a NDN client) must exchange different messages in 3 steps to retrieve HTTP content. First, iGW sends an *Interest* which name components contain, as illustrated in Table 1.a:

- the requested domain splitted by sub-domains and in reverse order (for example: www.google.com becomes /com/google/www),
- the path of the content on the web server,
- a hash of the request's header (a SHA1 of the HTTP header and up to 1024 bytes of the request body),
- the full route to the sender as a single name component in a binary TLV format.

This *Interest* packet is sent in the NDN network to ask someone to handle the request. So, the Egress gateway (or a NDN server) knows uppon reception that someone has an HTTP request to be satisfied, but also the netwok name to reach it. Please note that the SHA1 of the HTTP header is necessary to be sure that we respect the way the HTTP request was made to match users' properties (user agent, etc.). Choosing carefully a subset of fields to be considered to compute the hash can improve ICN caching while giving consistent results to users.

3) Retreive the HTTP request. Then, the Egress gateway extracts information from the first *Interest* sent by the Ingress gateway, more precisely the two last components: the sender route and the hash, and send an *Interest* in order to retrieve the full HTTP request (Table 1.b). Once the full HTTP request is received by the Egress gateway from the NDN network, it can now ask the HTTP server in the IP network for the actual web content.

4) Publish the HTTP response. After receiving the HTTP response form the IP web server, the Egress gateway splits it into *Data* packets with a NDN name like the first *Interest*, but without the sender route (Table 1.c). Following NDN principle, it is up to the NDN client (the original one or any other) to send *Interest* packets to retrieve each chunk of the HTTP response. In Table 2 is given an example of our naming pattern for the web content: "http://detectportal.firefox.com/success.txt".

IV. Implementation and Functionnal Validation

In order to validate the capability of our solution, we have conducted a set of functional tests which consist in determining the set of HTTP objects able to successfully cross our NFV-based NDN domain. In this section, we first present the architecture we have implemented. Then, we describe the scenario, tools and the dataset we have used to perform the tests. Finally, we provide the set of results we obtained as well as a brief related analysis.

A. Implementation and Testbed Deployment

Our testbed, depicted in Figure 1, is bi-located in two distant cities. Each site operates a set of servers and a VPN tunnel interconnects both. Following the NFV philosophy, the infrastructure is composed of standard x86 servers with the following configuration: dual Xeon CPU (E5-2630v3, 8c@2.4GHz), 128GB of RAM (DDR4) and 2x10Gbps (Intel X540) + 4x1Gbps (BCM 5720) for network connectivity. The VNF composing the logical architecture are instantiated by Docker (v1.6.0) containers based on Ubuntu server (v15.04). We can distinguish three types of components: NDN nodes executing a NDN Forwarding Daemon (NFD v0.5.1), the Ingress gateway and the Egress gateway. These two last components rely on the ndn-cxx (0.5.1) library. Finally, OpenVSwitch (v2.3.1) interconnects the different containers.

B. Dataset and Scenario

The first validation test our architecture has to pass, in order to demonstrate his capability to carry real end-users traffic accessing web content, consists in determining to what extent it is able to (1) successfully map HTTP requests and responses to respectively NDN *Interest* and *Data* packets and (2) transport this traffic across the testbed in a reliable way (i.e. without any loss of packet or excessive transit duration leading to the expiration of timers in HTTP user-agents). To that aim, we have challenged our architecture against the request of the top 1000 most popular HTTP web sites[2]. We deliberately do not consider HTTPS in this first version since HTTPS contents are end-to-end encrypted and thus cannot be converted by intermediaries (to benefit from ICN features like caching and multicast) if no agreement between the gateways and the endpoints to delegate keys are established. Each entry in this top-1000 consists in a web page standing for the entry point of the site and gathers a set of HTTP objects (e.g. images, audio, JavaScript, CSS sheets, etc.) which leads to several requests and answers to retrieve the whole page content. All websites were tested reachable but may still contain dead links. As a source request injector as well as response collector and parser, we have implemented a Web Scrapper based on the Jaunt implementation.

[2] https://gtmetrix.com/top1000.html

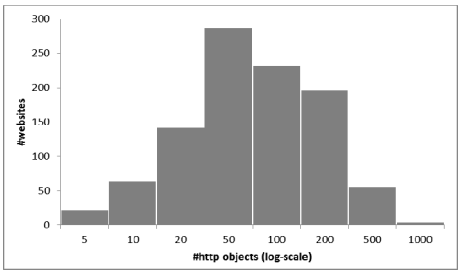

Figure 2: Distribution of content size of web pages, given in #web objects (semi-log scale)

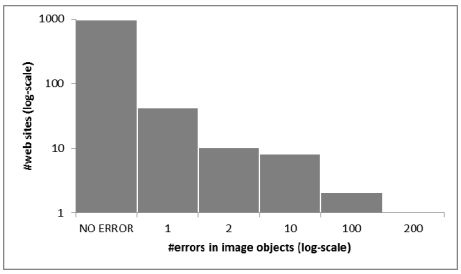

Figure 3: Frequency distribution of image errors (log-log scale)

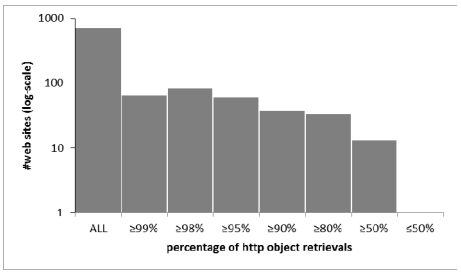

Figure 4: Frequency distribution of the percentage of successful web site retrieval (semi-log scale)

The latter especially provides insights about (1) connection errors (e.g. timeout) and (2) HTTP-related errors, which allows us to understand when and why our testbed may fail to carry some HTTP traffic. For each of these two cases, we relate the feedback collected and analyzed from the scrapper to a particular HTTP objet to eventually determine the capability of our architecture to carry such type of content.

C. Results

We have collected 73,236 web objects in total, whose distribution per site is provided in Figure 2, each retrieved with a dedicated HTTP request. Apart this set, we note that still 5,874 objects were not retrieved though the NDN island due to the use of HTTPS as a transport protocol. In this case, they are retrieved directly over IP (HTTPS traffic does not use the NDN network as web proxy) and we remove them from the set of results considered below.

As a first result, it appears that the testbed infrastructure has been able to retrieve 98.24% of all HTTP objects. Among these objects, 73.96% stands for images, 1.51% are scripts and 12.20% are links toward external resources (e.g. CSS sheets) while the last 2.33% are heterogeneous external resources embedded in tags such as iframe, area, span, etc. In order to be well understood, this global functional result must be put in front of the set of errors. We have shown in Figure 3 (plot in log-log scale for a better readability) that only a few web sites, hosting a significant set of image objects (from 10 to 100) induce most of the errors in this category while the majority of web sites gets a perfect or almost retrieval, thus assessing the global good performance of the gateways. Finally, we plot in Figure 4 the frequency distribution of the percentage of a web page content, given by the number of HTTP objects retrieved over the set of all objects that the infrastructure requests. The result (plot in semi-log scale) shows that most of top-1000 HTTP web sites can be retrieved entirely or almost while only a few (i.e. 17 sites over the 1000 considered in this study) bring poor results (i.e. less than 80% of content retrieval). This last category of sites is mainly composed of distant websites (Chinese, Korean and Russian), leading us to the conclusion that the problem comes from our eGW timeout value which is too restrictive for those particular websites. Concerning the overhead, the gateway adds approximately 30ms (only for the 1st packet, the rest of the content is streamed). The achieved throughput of 180Mbit/s is also good and in fact limited by the NDN routing function.

V. Conclusion and Future Work

NDN is a promising technology but its clean slate approach makes a large-scale deployment complex and unlikely. However, coupling NDN with a NFV approach can provide network operators with a progressive and affordable deployment solution, but it must be assessed before. As a first step, we built a testbed with the objective to process unencrypted web traffic in a virtualized NDN architecture based on Docker and OpenVSwitch. At the core of this infrastructure, we designed and implemented an HTTP/NDN gateway, that can map the HTTP protocol with NDN messages and properly deliver most of web content (>98%), according to our measurements. This allows ISP to progressively deploy ICN islands to benefit from this network stack features for content delivery in their core network, with the limited investment costs offered by NFV.

In our future work, we will improve the caching strategy of static web objects by giving them an identifier that can be shared between users and we will evaluate the gain of NDN when carrying web traffic. In parallel, we will develop the MANO part of our architecture by leveraging the TOSCA language to describe the network topology and to orchestrate the NDN-based VNFs. Our orchestrator will address in particular the scalability and the security of the NDN island, for instance by defining reaction policies to well-known NDN attacks like the Content Poisoning Attack, or to scale up the infrastructure when a gateway is overloaded.

References

[1] ETSI Group Specification. "Network Functions Virtualisation (NFV); Management and Orchestration". ETSI GS NFV-MAN 001 V1.1.1 2014-12.

[2] X. Ge, Y. Liu, D. H.C. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, and X. Hu. 2014. OpenANFV: accelerating network function virtualization with a consolidated framework in OpenStack. In *Proceedings of the 2014 ACM conference on SIGCOMM* (SIGCOMM'14). ACM, New York, NY, USA, 353-354.

[3] J. Soares, M. Dias, J. Carapinha, B. Parreira and S. Sargento, "Cloud4NFV: A platform for Virtual Network Functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on* , vol., no., pp.288-293, 8-10 Oct. 2014.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. 2008. OpenFlow: enabling

This work is partially co-funded by (1) the French National Research Agency (ANR), DOCTOR project, <ANR-14- CE28-0001>, started in 01/12/2014 and supported by the French Systematic cluster and (2) the CRCA and FEDER CyberSec Platform, <D201304601>.

innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (March 2008), 69-74.

[5] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri. Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the Ofelia testbed. Computer Networks, 57(16):3207-3221, 2013.

[6] X. N. Nguyen, D. Saucez, and T. Turletti. Providing CCN functionalities over openflow switches [Research Report] 2013. <hal-00920554>. 2013.

[7] N.L.M. van Adrichem and F.A. Kuipers, "NDNFlow: Software-defined Named Data Networking," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on* , vol., no., pp.1-5, 13-17 April 2015.

[8] M. Vahlenkamp, F. Schneider, D. Kutscher, and J. Seedorf. Enabling information centric networking in IP networks using SDN. In Future Networks and Services (SDN4FNS), pages 1-6, IEEE 2013.

[9] I. Moiseenko, D. Oran, "TCP/ICN: Carrying tcp over content centric and named data networks, in: *2016 conference on 3rd ACM Conference on Information-Centric Networking*, ACM, 2016, pp. 112–121.

[10] X. Marchal, M. E. Aoun, B. Mathieu, W. Mallouli, T. Cholez, G. Doyen, P. Truong, A. Ploix, E. M. de Oca, A virtualized and monitored NDN infrastructure featuring a NDN/HTTP gateway, in: Proceedings of the 3rd ACM Conference on Information-Centric Networking, ICN '16, Kyoto, Japan, ACM, 2016, pp. 225–226.

[11] F. Aguessy, O.Bettan, T. Cholez, G. Doyen, C. Enclos, H. Mai, W. Mallouli, X. Marchal, B. Mathieu, E. Montes de Oca, A. Ortiz, A. Ploix, P. Truong, "Architecture of the DOCTOR Virtualized Node", Technical Report, December 2015. http://www.doctor-project.org/outcome/deliverable/DOCTOR-D1.2.pdf

[12] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K.C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named Data networking. *SIGCOMM Comput. Commun. Rev.* 44, 3, 66-73. 2014

[13] S. Wang, J. Bi, J. Wu, X. Yang, and L. Fan, "On adapting http protocol to content centric networking," in Proceedings of the 7th International Conference on Future Internet Technologies. ACM, 2012,pp. 1–6.

[14] K. Ueda, K. Yokota, J. Kurihara and A. Tagami, "Towards the NFVI-Assisted ICN: Integrating ICN Forwarding into the Virtualization Infrastructure," *GLOBECOM 2016*, Washington, DC, 2016, pp. 1-6.