Enabling Native Coexistence Between ICN and TCP/IP Architectures Over the Same Domain

Eduardo Castilho Rosa[1], **Flávio de Oliveira Silva**[2]

[1]Department of Computer Science – Goiano Federal Institute (IFGoiano)
Catalão, GO – Brazil

[2]Faculty of Computing – Federal University of Uberlândia (UFU)
Uberlândia, MG – Brazil.

eduardo.rosa@ifgoiano.edu.br, flavio@ufu.br

Abstract. *Information-Centric Networking (ICN) has emerged as an alternative to overcome some issues imposed by TCP/IP architecture, such as the lack of mobility, security, and Quality-of-Service (QoS) native support. Seeing as TCP/IP will not disappear anytime soon, given the size of today's Internet, mechanisms to coexist multiples ICN architectures along with TCP/IP are necessary. Through concepts like Network Function Virtualization (NFV), Software-Defined Networking (SDN) and Data Plane Programmability (DPP), we propose in this paper the FIACS, a system capable of allowing pairs of entities to communicate transparently with each other by using its L2 access link and native protocols stacks over the same underline infrastructure.*

1. Introduction

Over the last few decades, the Internet has shown its resiliency by supporting features that were not present in the original design of TCP/IP protocols. Security mechanisms such as SSL/TLS and Virtual Private Network (VPN), as well as Content Data Networks (CDN), are some examples. Despite all the efforts to make the Internet more suitable for modern applications, the growing number of new use cases such as Vehicle-to-Vehicle (V2V) communication and the Internet of Things (IoT), create new challenges to the current TCP/IP architecture. In this sense, it is getting more and more necessary to rethink the original concepts behind TCP/IP to better support such applications. Among several clean-slate architectures to address some of the TCP/IP issues, known as Future Internet Architectures (FIA), those based on Information-Centric Networking (ICN) paradigm seems to be predominant in the literature according to [Doyen et al. 2019].

Overall, the basic principle behind an ICN architecture is to treat data as a first-class citizen and independent of its physical location. In-network caching mechanisms are used to ensure such property, distributing copies of data throughout the network, and allowing users to get the nearest available copy of data in the router's cache system. Thus, while TCP/IP forces the use of IP addresses to get the data, due to its host-to-host communication model, in ICN, the data itself is reachable from anywhere in the network through its unique name. ICN does not require previous knowledge of the information's location. This data-driven communication model is capable of better support some of today's applications as well as the futures ones.

There are several ICN architectures available in the literature, each one more appropriate for a particular use case. When it comes to mobility, for instance, the architectures MobilityFirst [Chen and Raychaudhuri 2010], PURSUIT [Lagutin et al. 2010] and CONET [Blefari Melazzi 2010] appears as good alternatives. Regarding IoT applications, some other architectures provide better support such as NDN (Named Data Networking) [Zhang and Claffy 2010] and GreenICN [TAGAMI and Arumaithurai 2016]. These architectures have features that enable them to be deployed at large scale without to need to use current Internet protocols and legacy equipment. So, the adoption of one particular ICN architecture to substitute the TCP/IP will not happen anytime soon, mainly due to the size of today's Internet. Thus, ICN deployments probably will have to coexist with TCP/IP for years to come, following the same road-map we have seen in both IPv4/IPv6 and 3G/4G transitions, for example.

A wildly adopted method to deal with the problem of ICN and TCP/IP coexistence is based on overlay approaches. In such a technique, the IP protocol is used to carry ICN traffic from one entity to another. This method is pretty simple and does not require any changes in the core network. Section 2 presents some overlay approaches to coexist ICN and TCP/IP. However, in the overlay technique is not possible to keep all the ICN native features such as caching, name-based forwarding, and routing as well as security and mobility. To overcome such limitations, we propose in this paper the FIACS (Future Internet Architectures Convergence System), which is a system capable of providing native end-to-end connectivity between ICN entities within a specific administrative domain.

We can deploy FIACS at Local Area Network (LAN), Internet Service Providers (ISP), data centers, or any other network in which the main goal is enabling traffic from multiples architectures independently of TCP/IP stack. FIACS provides an abstraction that allows any particular ICN or TCP/IP entity to exchange data with its peer over the same underlying infrastructure. FIACS is independent of the link access technology (such as optics fiber, phone cables, UTP) and L2 protocols (for example, PPP, Ethernet, or WIFI). Instead of using an overlay approach to carry ICN traffic, we propose a new encapsulation method that is generic enough to enable name-based forwarding in the data plane. FIACS makes use of technologies like Software Defined Networking (SDN), Network Function Virtualization (NFV), and Data Plane Programmability (DPP), as we will present in more detail in Section 3.

The rest of this paper is organized as follows: In Section 2, we present some related works highlighting its main features. In Section 3, we provide an overview of the FIACS's main components and how we can deploy it in real scenarios. Finally, in Section 4, we present some considerations and future directions.

2. Related Work

In [Ren et al. 2014], the authors proposed an ICN and TCP/IP coexistence solution based on SDN and NFV principles. The Versatile ICN Framework (VICN), as they called, can manage virtual ICN instances on top of VICN-enabled switches, ensuring properties like interoperability, routing, and security. In a way, VICN is similar to FIACS, seeing as both of them make use of a new kind of switch that is capable of forwarding different ICN traffic. VICN routers, though, are used to share storage capacity among several ICN instances entirely in a software layer to ensure features like in-network caching. In

FIACS, on the other hand, the ICN instances are distributed between the data plane, local control plane, and a remote control plane.

The authors in [Marchal et al. 2018] proposed an NFV architecture that enables the communication between HTTP entities over ICN islands. A gateway is introduced in this architecture to convert HTTP messages into ICN PDUs to become possible the interoperability between TCP/IP and ICN. Following the same idea, [Aguiar et al. 2019] also proposes an interoperation mechanism between ICN and TCP/IP, called FIFu (Future Internet Fusion). Differently from both works, however, FIACS focuses on just coexistence instead of interoperability and makes uses of SDN, NFV, and DPP all together.

Along the lines of using the DPP principle, the authors in [State et al. 2016] introduced the use of P4 language to treat native NDN traffic. Although P4 can brings many benefits to ICN, the author concluded that P4 has limitations in terms of dealing with text-based PDUs. FIACS can overcome such limitations by using IPC converters. Following the same principle, [Gavazza et al. 2019] presents a P4 switch capable of processing traffic from different ICN architectures. However, they do not show how text-based PDUs are processed in the data plane.

3. FIACS overview

The FIACS system is an alternative to overlay approaches that is capable of multiplexing traffic from different ICN and TCP/IP architectures under the same substrate. The idea is enabling entities with the same architecture to communicate to one another through a given path in which, from the entity's point of view, the flows are going through dedicated network equipment compatible with its architectures, when in reality, the traffic is crossing a shared core. This method avoids the need to have specifics equipment for each architecture that we want to support. Figure 1b shows the main components of the FIACS system. Each entity of a particular architecture can connect to FIACS by using a generic link technology where the native ICN or TCP/IP PDUs are encapsulated in propper L2 protocols associated with the corresponding link.

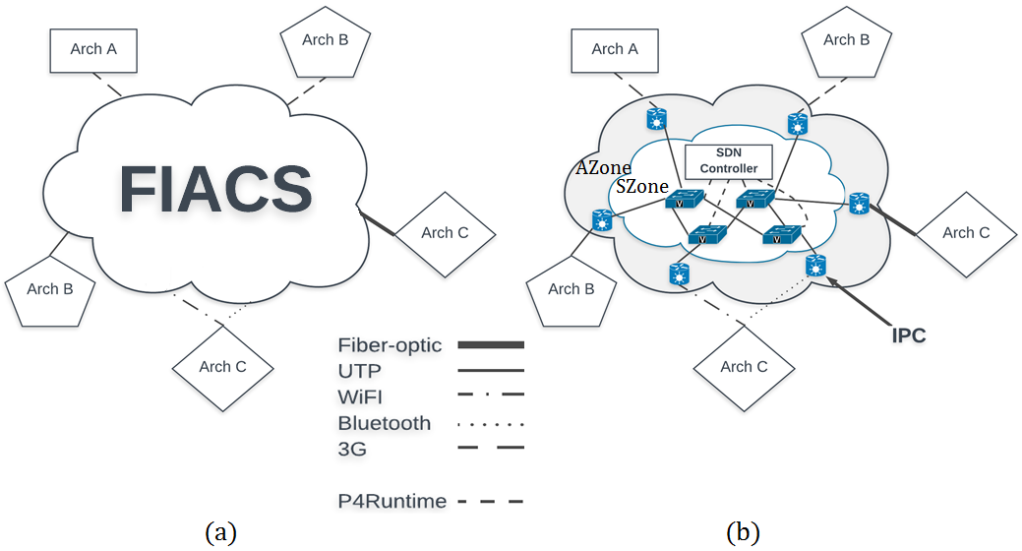

Figure 1. (a) Abstraction provided by FIACS and (b) all its main components.

As we can see in Figure 1a, each entity connects to FIACS using its available L2 access technology. An entity belonging to an architecture C, for instance, can connect to FIACS through different standards such as WiFI and Bluetooth, enabling multihoming, a popular feature in ICN architectures such as NDN. To do so, the FIACS has on its outer layer called Adaptation Zone (AZone), a set of network equipment called ICN PDU Converter (IPC) that performs the adaptation of the input traffic from a PDU format to a specific one, as seen in Figure 2. This specific PDU can be easily forwarded in the inner layer, called SDN Zone (SZone), regardless of the natives ICN PDUs format.

One crucial role that IPC plays in FIACS is performing the L2 signaling protocols and link termination. For example, in a WIFI connection, the IPC is responsible for manage the exchange signaling messages and authentication mechanism. After that, every incoming packet received by an input interface is encapsulated and sent it out to SZone to be natively forwarded, as we will see next in this work. Specifying a new encapsulation method turns the switches in SZone capable of processing packets based on its binary fields, although ICN architectures are name-based. This encapsulation facilitates the processing of ICN packets at line rate enabling features like long-prefix matches even using named-base forwarding. However, one drawback of using this encapsulation is the extra overhead generated for each packet. We have been conducted some experiments to evaluate the impact of such overhead and we will present it in future works.

The encapsulation that IPC does in every incoming packet aims to adapt the ICN traffic at the edge network to a common PDU format that carries all possible ICN PDU formats as well as TCP/IP in the core network (SDZone). Figure 2 shows the PDU format that IPC can generate. The type field is responsible for identifying the architecture correspondent to that payload. There are two possibilities here: 1) payload belongs to an architecture that generates text-based PDUs like NDN, and 2) payload belongs to an architecture that generates binary-based PDUs like TCP/IP in its L2-L4 protocols. In the former case, we need to take all the name components, for instance, the name of a particular data and hash them using a specific algorithm given by hash type field code. To clarify, lets suppose we take a NDN name called */ufu/facom/mehar/presentation/p1*. In such a case, we set the number blocks field to five (number of blocks), and then we hash each name part individually using a specific algorithm and put all hashes into the next five subsequent fields in which size is hash-type dependent. At the end of all hashed blocks, we specify what type of text encoding algorithm we have used to encode the payload. To ensure the integrity of the entire PDU, we calculate the CRC and appends it at the end of the packet. In the latter case, for architectures that produce binary-based PDUs, we set both hash type and number blocks fields to zero and skip to the encoding field where we perform the same operations as we do in the previous case.

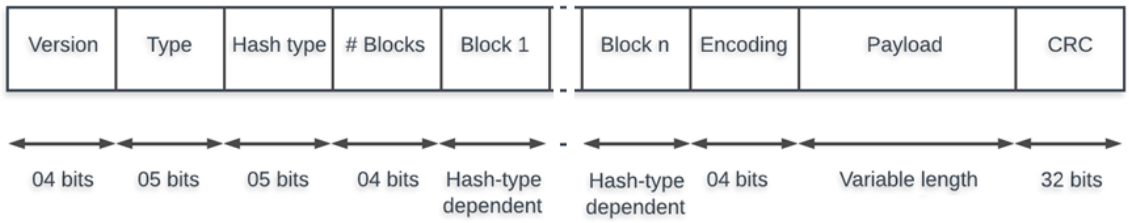

Figure 2. FIACS PDU format.

All incoming traffic processed by an IPC is sent it out to SZone. This zone represents a fabric that interconnects multiples switches in a Clos topology. The advantage of this topology is that it can scale up easily just by adding new switches as we need, becoming possible to increase the number of entities at any time. We propose a switch model called *cFlex* that we can implement it by using server with a customized hardware like FPGAs or other P4-capable SmartNics such as Agilio CX [Netronome 2018]. Other than the local data plane and remote control plane that we have in traditional SDN switches, our *cFlex* switch model adds the concept of a local control plane that runs on top of a programmable ASIC or FPGAs. That local control plane includes the Network Operating System (NOS) and a set of VNFs deployed on top of it. We can implement each VNF either as a virtual machine or a docker container and represents a set of features of a given architecture. To get rid of TCP/IP stack in VNFs, we bypass the NOS and uses raw sockets to receive and transmit customized PDUs directly into the ASIC or FPGA using the PCI-E bus. Thus, as we can see in Figure 3, each architecture supported by FIACS has its components distributed in the local data plane, local control plane, and remote control plane. By using this approach, each architecture runs in a separate slice that can be instantiated automatically or manually. So far, we are using the latter case, but at the same time, we are investigating the possibility of using a system capable of orchestrating such slices automatically.

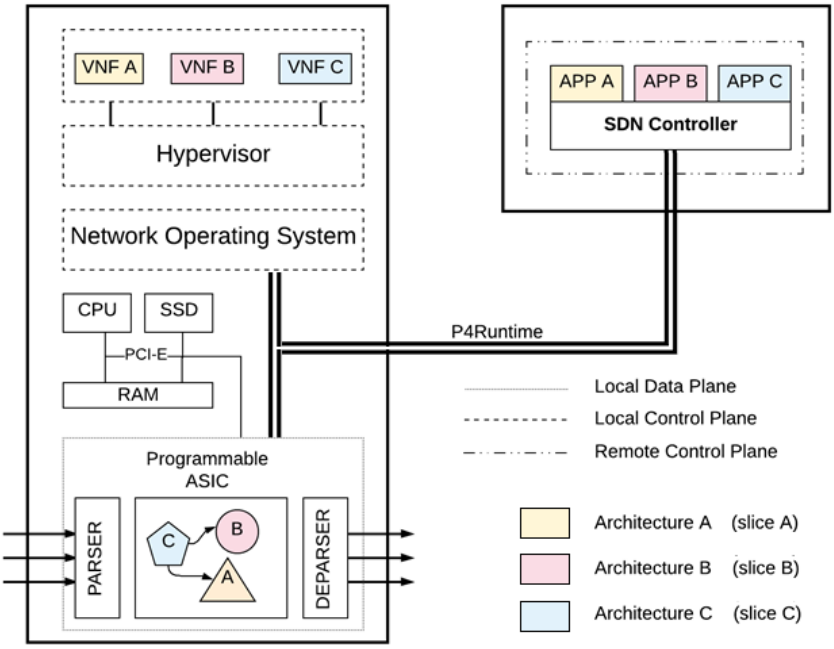

Figure 3. Proposed *cFlex* switch model.

The arriving packets in *cFlex* are initially processed in local data plane by using a P4 program. At least for now, IPC does not support the P4 language and all the traffic generated by it uses our own packet process engine. We uses P4 to implement a customized pipeline with specific tables and actions for each architecture. Of course, not every feature of a given architecture can be expressed in P4 and processed fast in the data plane. Due to the TCAM's memory limitations, for example, in-network caching is not possible in the data plane. To deal with this, for each feature of a given architecture that we can not implement it in P4, we package such feature as a VNF and get it executed in the local

control plane using a general-purpose CPU. In other words, what we are doing here is offloading some functions from the local control plane (slow path) to the data plane (fast path). However, by doing so, we may add a bottleneck at the PCI-E bus depending on how often we send packets to be processed in VNFs, which can cause some degradation in terms of latency and bandwidth. We have been conducting some experiments to evaluate, for a given number of architectures, how often the flows in the data plane needs to be processed in VNFs at local control plane and the impact of it in terms of bandwidth and latency. We use the P4Runtime API to enable the communication between the local data plane and the local control plane, as we can see in Figure 3.

Another essential component of our model is the remote control plane, which is deployable as a SDN controller. For each architecture that FIACS supports, we have a particular SDN application that is responsible for establishing the flow path interconnecting peers of entities that are exchanging data to one another in a given moment. All these paths are created dynamically by populating the tables in the pipeline using the P4Runtime API. Besides the applications for each architecture, we also implement in the SDN controller, an application to VNF placement across all *cFlex* switches in SZone. All VNF images files are previously-stored at each *cFlex* switch, and the corresponding SDN applications will perform their instantiation.

4. Considerations and Future Directions

We presented in this work FIACS, a system to enable the coexistence between multiples ICN architectures as well as TCP/IP. The abstraction provided by FIACS allows entities to communicate with one another transparently using its native architectures over L2 access technology available to them. The combination of emerging technology such as SDN, NFV, and DPP facilitates the coexistence between ICN and TCP/IP improving the trade-off between performance and flexibility.

In the next steps in this work, we intend to include the development of all the components we have proposed here, such as IPC, *cFlex*, and the SDN controller. To demonstrate the feasibility of FIACS, we'll first get a proof-of-concept implementation by using low-cost devices like Raspberry Pi. Once we have promising results, we will focus on the performance aspect of FIACS by using servers with customized hardware like FPGAs or other P4-capable SmartNics such as Agilio CX [Netronome 2018]. By doing so, we expect that FIACS can bring many benefits in terms of low latency to applications running on top of ICN or TCP/IP architectures.

Some challenges imposed by FIACS so far include the need to have physical IPC converters for each group of entities that can not scale up so quickly. One other future direction is to virtualize the IPC by using commodity hardware. Thus, we can make one single virtual IPC capable of connecting multiples entities with different L2 access technology.

References

Aguiar, R. L., Guimarães, C., Quevedo, J., Ferreira, R., and Corujo, D. (2019). Exploring interoperability assessment for Future Internet Architectures roll out. *Journal of Network and Computer Applications*, 136:38–56.

Blefari Melazzi, N. (2010). The Convergence Project.

Chen, J. and Raychaudhuri, D. (2010). MobilityFirst Future Internet Architecture Overview.

Doyen, G., Cholez, T., Mallouli, W., Mathieu, B., Mai, H., Marchal, X., Kondo, D., Aouadj, M., Ploix, A., Montes-de Oca, E., and Foster, O. (2019). An Orchestrated NDN Virtual Infrastructure Transporting Web Traffic: Design, Implementation, and First Experiments with Real End Users. *IEEE Communications Magazine*, 57(6):33–39.

Gavazza, J. A., Santos, M., Jr, P. D. M., Verdi, F. L., and Monteiro, J. A. S. (2019). Implementação de um switch em P4 com suporte simultâneo a múltiplas arquiteturas de Internet do Futuro. *Anais do Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF)*, pages 13–18. Conference Name: Anais do X Workshop de Pesquisa Experimental da Internet do Futuro Publisher: SBC.

Lagutin, D., Fotiou, N., and Tsilopoulos, C. (2010). Publish Subscribe Internet Technology | PURSUIT Project | FP7 | CORDIS | European Commission.

Marchal, X., Aoun, M. E., Mathieu, B., Cholez, T., Doyen, G., Mallouli, W., and Festor, O. (2018). Leveraging NFV for the deployment of NDN: Application to HTTP traffic transport. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5.

Netronome (2018). Agilio CX SmartNICs. Library Catalog: www.netronome.com.

Ren, J., Lu, K., Wang, S., Wang, X., Xu, S., Li, L., and Liu, S. (2014). VICN: a versatile deployment framework for information-centric networks. *IEEE Network*, 28(3):26–34.

State, R., Signorello, S., François, J., and Festor, O. (2016). NDN.p4: Programming information-centric data-planes. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 384–389. ISSN: null.

TAGAMI, A. and Arumaithurai, M. (2016). GreenICN Project: Architecture and Applications of Green Information Centric Networking. *IEICE Transactions on Communications*, E99.B:2470–2476.

Zhang, L. and Claffy, K. (2010). Named Data Networking (NDN) - A Future Internet Architecture.