Toward Next-Generation and Service-Defined Networks: A NovaGenesis Control Agent for Future Internet Exchange Point

Thiago Bueno da Silva, Fábio L. Verdi, Juliano Coelho Gonçalves de Melo, José Augusto Suruagy, Flávio Silva, and Antônio M. Alberti

Abstract

The Internet has modified how we interact with the world, disrupting our leisure, work model, and economy. Nonetheless, it presents an ossified architecture and infrastructure that may burden future applications. These limitations have led to the development of future Internet architectures (FIAs), for example, NovaGenesis (NG). Considering recent trend topics such as next-generation networks, software-defined networking, service virtualization, and 5G network slicing, the coexistence of disjointed architectures in a multi-architectural network is the key to supporting applications with diverse requirements. In this article, we present NG data and control planes for a programming protocol-independent packet processor (P4)-based future Internet exchange point (FIXP). This proposal advances the state of the art by delivering a native and efficient NG controller that manages a multi-architecture exchange point on the fly. In this article, the NG controller prototype orchestrates a simulated autonomous system with FIXP to connect NG hosts (and its content distribution application) dynamically. The obtained results validate the solution.

Introduction

The Internet is an essential artifact for the contemporary world. Over a half century, this infrastructure has expanded exponentially to provide access to the most varied services, disrupting the economy and connecting several sectors to exchange essential data. In particular, the current global panorama has relied even more on the Internet. Based on some surveys, the Internet's demand and data traffic increased as the world adapted to this new remote scenario. Unsurprisingly, the complaints about Internet quality have also intensified.

This is not an outlier. The Internet has been the cornerstone of the Internet of Things (IoT) [1], enabling ordinary objects to exchange data based on sensors and actuators. Through a myriad of connected devices, smart-* proposals aim to disrupt how we interact with the world. As examples, we can combine trends such as smart cities, autonomous vehicles, and wearables to create a digital twin from a person to an urban ecosystem. Moreover, over-the-top (OTT) providers, microservice architectures, and legacy applications exploit this infrastructure to offer the most diverse services [2]. Nonetheless, they all rely on the Internet, an ossified architecture and infrastructure that has struggled with well-known limitations.

Concerning the Internet evolution, there are two approaches to create future Internet architectures (FIAs). On one side, evolutionary efforts optimize the network without discarding the TCP/IP architecture (e.g., the shift from IPv4 to IPv6). Nevertheless, some say these proposals will never solve Internet conceptual problems, such as mobility, naming, and named data routing. Therefore, revolutionary efforts rethink the Internet from a contemporary perspective, integrating the state-of-the-art paradigms to conceive clean-slate architectures [3]. These proposals consider the lessons learned from the current Internet and address its limitations.

Therefore, these revolutionary proposals might enhance novel technologies, OTT providers, and microservices under new paradigms. For instance, named data networks (NDNs) [4] and publish-subscribe Internet technology (PURSUIT) [4] are types of information-centric networking (ICN) that have content as the cornerstone and present a stateful routing scheme based on the content's name resolution. Recursive Internet architecture (RINA) [5] claims that computer networks are inter-process communication, so it provides a single-layer abstraction to be recursively distributed over the network. Meanwhile, entity title architecture (ETArch) [6] establishes workspaces to meet quality of service, mobility, and security while supporting dynamic service allocation for its applications.

Motivation

Given this landscape, the future Internet can present several disjointed architectures coexisting in the same network infrastructure, sharing the available resources to fulfill future applications and services. As these proposals do not support the same principles, it is a challenge to enable this environment. A future Internet exchange point (FIXP) can promote a multi-architectural Internet and exchange their data through programmable hardware. Moreover, the whole Internet ecosys-

Thiago Bueno da Silva and Antônio M. Alberti are with Instituto Nacional de Telecomunicações, Brazil; Fábio L. Verdi is with Federal University of São Carlos Sorocaba, Brazil; Juliano Coelho Gonçalves de Melo and Flávio Silva are with Federal University of Uberlândia, Brazil; José Augusto Suruagy is with Universidade Federal de Pernambuco, Brazil.

Digital Object Identifier:
10.1109/MNET.008.2100555

tem can benefit and profit from this by increasing the services' diversity with better infrastructural support [2]. Nevertheless, we need to decouple policies from technologies on Internet governance, allowing alternative architectures to be offered globally without missing the Internet role for humanity. FIAs can evolve or revolutionize Internet technologies while still complying with globally established Internet policies.

Software-defined networking (SDN) fosters this multi-architectural Internet by leveraging reprogrammable hardware [7]. Upon segregating the data and control planes, network providers can simplify their infrastructure, support its evolution natively, and optimize its overall service. This trend has grown by the convergence of SDN and next-generation networks (NGNs) while also enabling FIA deployment. It is also a hot topic for 5G and post-5G mobile networks [8].

This work extends recent research that propose the NovaGenesis (NG; https://github.com/antonioalberti/novagenesis/) data and control planes integration through a programmable FIXP with new features and improvements. NG presents several common features, such as flat identifiers, mobility, and name resolution, and unique concepts in the FIA landscape, such as protocol implementation as services, contract-based operation, and semantics-rich self-organization. Upon selecting NG, we validate the design of a native SDN controller based on a representative FIA. Our solution supports NG novelties natively, including all these features, and it contrasts with the related works that present external controllers.

Research Contributions

The main contributions of this article are the following:

- Design of an innovative and native NG SDN/NGN controller for the FIXP programmable Data Plane. To the best of our knowledge, this is the first FIA-native SDN controller developed in the literature.
- Performance evaluation of the native NG controller, the FIXP virtualized infrastructure, and the host machine. This work presents the obtained results to prove the feasibility and effectiveness of our proposal.
- It provides an unprecedented discussion about the design of SDN/NGN for FIAs, covering the FIXP methodology and comparing it to some related works.

Article Organization

The remainder of the article is organized as follows. The next section provides a brief background discussion. Then we cover related works in terms of FIA and SDN, analyzing the current state of the art. Following that, we highlight the convergence between FIXP and NG, exploring the essential aspects to enable this proposal. Then we present the proposed methodology to validate the performance and the obtained results. The final section concludes this article.

Internet Evolution

This section focuses on the Internet evolution, encompassing a brief perspective of SDN and NG as a FIA proposal.

Concerning the Internet evolution, there are two approaches to create future Internet architectures (FIAs). On one side, evolutionary efforts optimize the network without discarding the TCP/IP architecture (e.g., the shift from IPv4 to IPv6). On the other side, revolutionary efforts rethink the Internet from a contemporary perspective, integrating the state-of-the-art paradigms to conceive clean-slate architectures.

SDN and P4

By proposing programmable hardware, SDN fosters the network infrastructure flexibility. Because of this paradigm, the equipment becomes modeled by software, allowing the network applications to evolve through reprogrammable devices [7]. Hence, it segregates the data traffic into well-defined planes, in which the data plane confines the service's data, the control plane ensures the control of the infrastructure, and the application plane fosters the network management. Through this disruption, the network becomes logically centralized and horizontal, wherein network applications update controllers' knowledge to configure the forwarding devices dynamically.

Through this harmony, SDN ensures the evolution of networks, optimizing decision making and providing efficient management. In addition, it also helps to reduce the capital and operational expenditure of Internet providers once the infrastructure can be automated [7]. When contemplating the future of the networks, SDN can foster autonomous networks with the required awareness to adapt accordingly to its dynamic needs and intent-based networks that orchestrate themselves based on high-level objectives. Finally, SDN has allowed a new level of experimentation with novel network proposals once researchers can now entirely model the device behavior.

Programming protocol-independent packet processor (P4) architecture is an SDN example that enables the network forwarding devices' programmability [9]. This proposal has addressed OpenFlow limitations, ensuring greater independence of protocols and devices by using the concept of protocol-independent switch architecture. Therefore, any P4 device becomes reconfigurable by describing its behavior, tables, and actions in the P4 program. In turn, the P4 compiler translates the P4 language into machine instructions for a network application. Even though this framework lacks support for native P4-based controllers, P4 runtime provides the required application programming interface (API) for supporting a customizable controller. Hence, any SDN controller can dynamically interact and manage the data plane.

NovaGenesis

NG is a convergent information architecture that redesigns the Internet, integrating the state of the art to foster data exchange, storage, and processing [10]. Protocols are implemented as services using a basic set of architectural components and design choices such as naming, name resolution, and life cycling. NG offers basic services for new protocol implementation (dynamic protocol stacking) and supports services' self-organization and composition. NG goes beyond traditional SDN, supporting a new paradigm named service-de-

Definition (acronym)	Description
Content application (ContentApp)	Specialized network service for content distribution. An NG host can be configured as a content source, establishing a contract to temporarily store its data, or a content repository, offering its storage capacity.
FIXP internal broker (FIB)	FIXP abstraction layer middleware that supports the FIXP control and data plane operations.
FIXP switch (FSW)	P4-based forwarding element at FIXP data plane.
Name resolution and networking cache service (NRNCS)	Basic service that promotes data exchange, name resolution, and network caching for name bindings and content objects.
NG control agent (NGCA)	An NG SDN controller capable of dynamically managing FIXP Data Plane on the fly.
NG core (NGCore)	An NG peer with NRNCS to resolve names and store temporary content improving overall network quality of service.
NG repository (NGRepo)	An NG peer with ContentApp role of Repository, which stores content for a long period off the network.
NG source (NGSSrc)	An NG peer with ContentApp role of Source, which presents some named content to be distributed.
Proxy/gateway/controller service (PGCS)	Basic service that acts as a gateway encapsulating the NG protocol into legacy link-layer technologies standards; as a proxy to represent physical devices; and as a controller to configure programmable devices.

TABLE 1. NG and FIXP key components.

fined networking, and network slices are made by establishing service contracts. This project integrates state-of-the-art ingredients to deliver an infrastructure that has supported IoT, software-defined radio, and Industry 4.0 natively.

When starting operating, services expose their names, which are natural language names or self-verifying names (SVNs) derived from a cryptographic hash function. These names can be identifiers, locators, semantic annotations, or metadata. Exposing messages are periodically broadcasted to peers, fostering service offers based on discovered trust relations (a crucial feature for all software that runs outside a blockchain). NG provides name- and contract-based service self-organization.

NG can be considered an NGN since it offers a set of expandable features for cooperative and distributed control and management. For example, the proxy/gateway/controller service (PGCS) is a basic service for software routing, encapsulating messages over legacy network protocols and making NG an overlay application. It also represents physical devices in the service ecosystem, acting as a gateway between software and network interfaces in an operating system. PGCS can represent a physical switch to enable hardware dynamic configuration based on established service contracts. Meanwhile, a specialized service content application (ContentApp) enables content exchange among data sources and repositories, based on PGCSs and name resolution and network cache service (NRNCS). User awareness is implemented in ContentApp, allowing self-organization of content distribution according to users' preferences. This network application addresses the problem of self-organizing content distribution with integrity and provenance. Table 1 briefly lays out the NG and some FIXP key components related to this work.

NG communication follows the publish/subscribe model. PGCS encapsulates messages to any link-layer protocol. Every message presents the routing line, establishing its source host identifiers (SHIDs) and destination host identifiers (DHIDs). These identifiers are the basis for the NG forwarding scheme with different routing strategies. Considering NGHello, it presents a tuple of "FFFFFFFF" as its DHIDs, characterizing a broadcast message. As PGCS can fragment messages to comply with the legacy technology, the routing line is usually present only at the first fragment. Meanwhile, subsequent packets only share the Message Identifier (MsgID) and Fragment Sequence (FragSeq) fields.

With these characteristics, NG fosters our proposal of developing an NG SDN controller for a multi-architecture ecosystem, exploiting its naming and name resolution features, as well as trustable service self-organization. Focusing on the SDN controller, the best option is to extend the current PGCS scope to orchestrate programmable hardware. This service has already been applied for representing IoT devices in a service-defined distributed control and management solution [10].

Software-Defined Networking Applied to the Future Internet Architectures

SDN has been popularized over proposals such as OpenFlow and P4. The concepts related to each proposed FIA is beyond the scope of this work. Hence, we focus on how each related work exploits SDN, narrowing to a P4 proposal when possible.

Enhanced NDN (ENDN) [11] explores NDN to fulfill its stateful routing scheme, overcoming the P4 limitations through an architecture capable of handling string-based content and managing network slices. For evaluation, an ndnSIM simulator created two different topologies with content consumers and producers, where ENDN controllers use P4 Runtime to manage Behavioral Model version 2 (BMv2) targets in the DP.

Focusing on optimizing recursive internetwork architecture (RINA) interior and border routers, Gimenez *et al.* developed a P4-based router [12]. The authors designed a switch capable of handling RINA's data transfer protocol with high-performance demands, such as low latency, high throughput, and flexibility. Moreover, they investigated throughput and packet losses through virtual machines (VMs).

Feng *et al*. [13] proposed a hybrid protocol that enabled HTTP in ICNs. The proposal developed P4-based switches capable of forwarding data from both protocols, wherein proxies translated the HTTP and ICN packets before transmitting them on the network. Through their evaluation, the authors fulfill their goal, reducing redundant data and optimizing the data exchange.

Guimaraes *et al*. [14] proposed an interoperation architecture for NDN, PURSUIT, and IP. Their architecture has two layers, in which gateways ensure the inter-communication and controllers to support the DP operation and performance. Through a virtualized network, the authors evaluated the performance in terms of fetching and processing time, consumed bandwidth, and messages forwarded per second.

Table 2 summarizes our analysis, presenting proposals that optimize FIAs or foster a multi-architecture network, enabling the interoperation of heterogeneous clients, that is, any client can access any content in the network. None of these works presents its control plane for a FIXP, focus-

Proposal	Approach	Architectures	Objective	Technologies	Network metrics
[11]	Revolutionary.	NDN.	Enable NDN's routing scheme with FIB, PIT, and CS over network slices.	ndnSIM and P4.	Latency and throughput.
[12]	Revolutionary.	RINA.	Optimize RINA's interior and border router for high-performance scenarios with SDN.	Mininet, AWS VMs, Stratum Network O.S., and P4.	Throughput and packet loss.
[13]	Revolutionary.	TCP/IP and ICN.	Decouple HTTP protocol from TCP/IP, enabling its communication scheme with ICN.	Ubuntu 16.04, Mininet, P4, and custom proxies.	Server response load.
[14]	Interoperation.	NDN, PURSUIT, and TCP/IP.	Create an inter-communication environment that enables clients to receive any available content.	VMs and custom software	Fetching time, computation overhead, bandwidth, and throughput.

TABLE 2. Comparison of SDN initiatives for future Internet architectures.

ing only on the data plane. Therefore, this work proposes an NG controller to manage a programmable multi-architecture FIXP data plane.

Future Internet exchange Point

This section presents the FIXP architecture and the NG forwarding device and controller. First, it focuses on an in-depth analysis of the FIXP elements for a multi-architecture environment. Then it highlights the design principles for fostering the development of NG's forwarding device and control agent. Finally, it characterizes the current prototype.

FIXP Architecture

FIXP is a programmable exchange point for multiple communication architectures, wherein different Internet providers enable their clients' traffic, leveraging a flexible infrastructure that supports their divergent services [15]. FIXP can be deployed as a point of presence, an Internet exchange point, or an overlay application as middleware for virtualized network functions and the underlying P4-based infrastructure. The current solution enables the coexistence of the current Internet (TCP/IP) and two FIA proposals (ETArch and NG), combining SDN and custom software through the FIXP protocol.

Figure 1 highlights the FIXP architecture that presents three layers. The topmost FIXP control layer (FCL) belongs to the FIXP control plane (FCP), containing heterogeneous and disjointed SDN controllers that provide high-level services to manage the network. The FIXP abstraction layer (FAL) represents the southbound interface of the FIXP protocol, abstracting the complexity of the FIXP data plane (FDP) and easing the network management. In this layer, the FIXP internal broker (FIB) is the middleware that supports FCP and FDP. At the bottom, the FIXP physical layer (FPL) is the infrastructure layer that encompasses the physical topology with P4-based FIXP switches (FSWs). These FSWs are programmable forwarding devices, wherein *FIXP.P4* models the P4 switches' behavior, and the FIXP Rule Handler Service (FRHS) provides the means necessary to configure the P4 forwarding tables.

Summarizing FIXP's internal flow, an FSW receives an Ethernet packet (1). *FIXP.P4* analyzes the received packet, deciding to request guidance from FCP for an unknown destination (2) or forward it to a known destination (12). In its turn, the FIB receives the FIXP Packet-in primitive (2) and sends this to its corresponding controller (3). Each FIXP SDN controller has its routing strategy to set the underlying FDP. These SDN controllers send FIXP control primitives (4) to modify its architecture's forwarding table, and the FIB forwards them to the desired FSW (5). Upon receiving this, FRHS translates it to the P4 Runtime standard (6), configuring the FSW. After that, the FSW generates the FIXP Acknowledge (FAck) feedback to the controller (7, 8, 9). If the status is successful, the controller can reinsert the data packet initially received in (1) so that this information is not lost. This reinsertion flow is represented in (10, 11, and 12). Hence, the set FSW can forward packets to the same destination without FCP intervention.

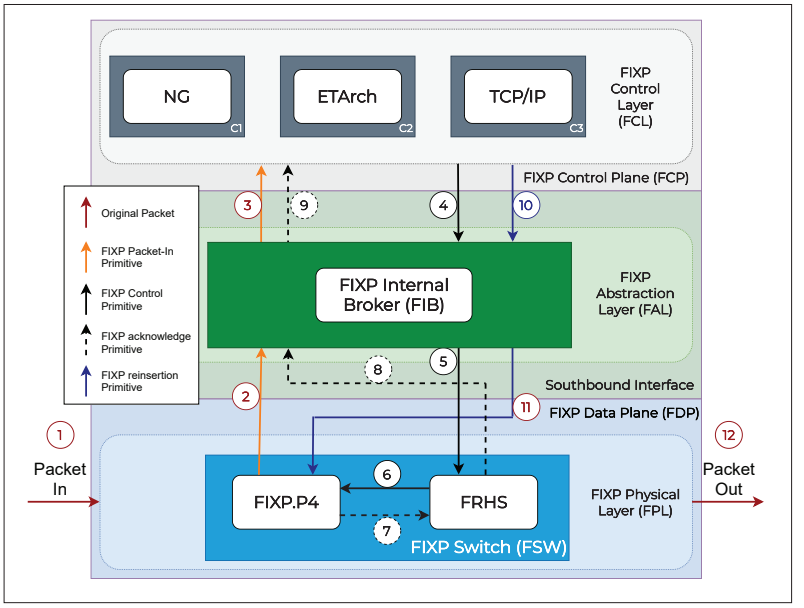

FIGURE 1. FIXP architecture, illustrating its internal communication structure.

NG Forwarding Devices and the NG Control Agent

A native NG control agent (NGCA) extends the PGCS, supporting NG features like semantics-rich self-organization, broadcast, and flat identifiers. Moreover, this SDN controller has followed a bottom-up design that orchestrates the FIXP programmable data plane on the fly. Figure 2 illustrates the sequence diagram for the NG packet processing in the FIXP architecture. This flowchart highlights the NGCA's actions.

Considering NG, the FSW forwards its packets based on their DHID (i.e., NG destination identifier(s)) (1). As NG peers periodically expose their resources through NGHello broadcast, FSW multicasts these packets to all known NG peers except for its source (2). Considering an unknown DHID (5), FSW sets the NG reserved bytes with its FSW identifier (FSWID) and the FSW ingress

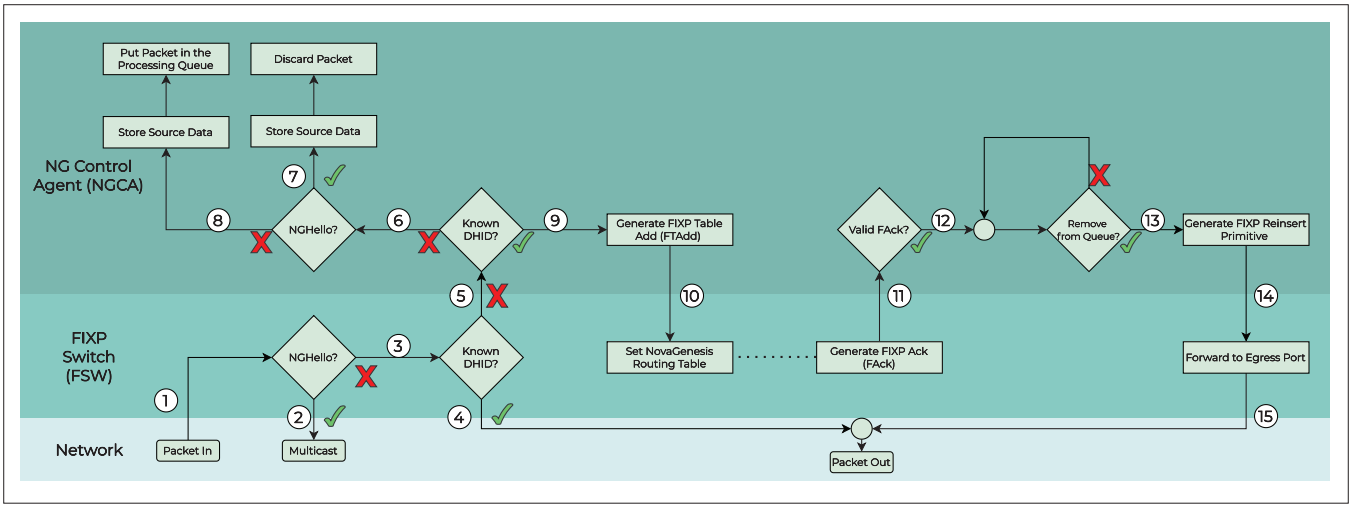

FIGURE 2. NG Controller and FIXP flowchart. This image depicts the interaction between the FDP and FCP to enable the NG data exchange. For any incoming NG packet, the FSW can unicast, multicast, or request guidance from the NGCA to forward messages based on their DHID.

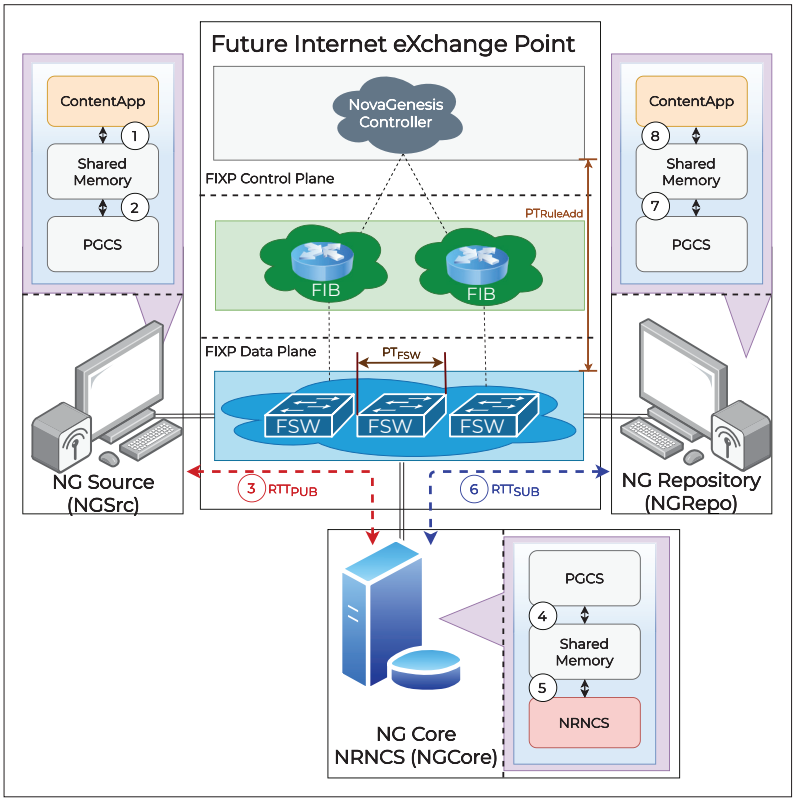

FIGURE 3. FIXP evaluation with NG content and distribution application scenario, wherein NGSrc, NGCore, and NGRepo exchange photo content.

port (FSWIP) upon requesting guidance from FCP. Based on them, NGCA can set the FDP network topology on the fly. For fragmented messages with no explicit DHID, FSW exploits registers to retrieve the host ID from a hash function with the first fragment's MsgID.

NGCA draws the following strategy. At first, it receives all the fragments to rebuild a message, analyzing whether it is an NGHello or not (6). In either case, NGCA registers the SHID, FSWIP, and FSWID into a database called FIXP Knowledge, abstracting the FDP dynamically. NGCA discards an NGHello (7), since it has already been multicasted, or inserts the different messages into a queue to later reinsert them (8).

When NGCA discovers how to reach a destination (9), it generates the FTAdd to set the requesting FSW (10). For a successful FAck (11), NGCA updates its database (12) and reinserts any packet with the same DHID at the set FSW (13, 14, and 15). From now on, the set FSW can forward any packet flow to the same DHID without the controller's guidance.

Current Prototype

The current FIXP prototype enables the coexistence of the TCP/IP, ETArch, and NG architectures. For NG, it supports NG features for service exposition, message fragmentation, and data exchange. NGCA is a controller that extends the native PGCS to orchestrate a dynamic FIXP network topology.

Proof of Concept

In this section, a VirtualBox virtualized environment emulates an Internet provider's autonomous system that connects three NG hosts through FIXP. Therefore, NGCA orchestrates the flexible P4 data planes to fulfill their communication.

Evaluation Scenario

For evaluation, an Internet provider connects three NG hosts (Fig. 3). These hosts perform the NG content distribution network application scenario, wherein one NG source (NGSrc) discovers one NG repository (NGRep) and one NG core NRNCS (NGCore) to store 1000 photos of 7.5 kB. After establishing a service contract, the NGSrc publishes photos from its ContentApp (1), exploiting its shared memory (2) to deliver them to its PGCS. Therefore, NGSrc's PGCS encapsulates the content into Ethernet packets to send to NGCore (3). Upon receiving the packets, NGCore's PGCS reassembles the photos (4) to temporarily store them into its NRNCS (5), depicting an ICN web concept. Hence, NGRep's ContentApp subscribes (6) to the closest available content source, that is, the NGCore. In this way, NGRep's ContentApp receives the photo content from its PGCS (7) and its domain shared

Topo.	RTT_{PUB}/RTT overhead	RTT_{SUB}/RTT overhead	PT_{FSW}	$PT_{RuleAdd}$	CPU usage/overhead	Memory usage/overhead
Case 1	126.7 ± 7.2ms/—	208.2 ± 10.3 ms/—	0	0	16.7/—	02.4/—
Case 2	130.5 ± 7.1ms/02.9%	215.6 ± 10.3ms /03.6%	8.7 ms ± 64.4 µs	114.9 ± 08.9 ms	20.3/1.7%	05.2/117.8%
Case 3	138.2 ± 7.3ms/09.1%	237.2 ± 10.8ms /13.9%	8.6 ms ± 55.3 µs	144.5 ± 23.6 ms	20.9/25.7%	05.9/147.7%
Case 4	140.7 ± 7.3ms/11.1%	225.2 ± 10.4ms /08.2%	5.9 ms ± 29.2 µs	137.5 ± 07.8 ms	22.1/32.3%	07.9/230.3%
Case 5	145.1 ± 7.3ms/14.4%	224.1 ± 10.2ms /07.6%	5.6 ms ± 25.7 µs	111.3 ± 02.7 ms	24.5/46.9%	09.5/292.5%
Case 6	151.6 ± 7.4ms/19.7%	239.5 ± 10.5ms/5.1%	3.6 ms ± 22.0 µs	122.4 ± 05.8 ms	25.6/52.3%	10.7/343.9%

TABLE 3. NovaGenesis control agent and FIXP performance results.

memory (8), storing only the unique content. It is important to point out that the focus is to validate the FIXP capacity of abstracting and supporting NG features in a reasonable time, so the size of the photos only impacts the number of packet fragments.

The evaluation considers six possible network topologies for comparing FIXP performance. Case 1 has a virtual network to interconnect NGSrc, NGRep, and NGCore, that is, a standard scenario without FIXP. The other five cases exploit FIXP with one NGCA orchestrating the autonomous system dynamically without human intervention. The main difference relates to the number of FSWs at FDP: Case 2 presents one, Case 3 deploys three, Case 4 uses five, Case 5 exploits seven, and Case 6 applies nine FSWs to forward data between NG clients. As each FIB abstracts up to five FSWs, Cases 2, 3, and 4 require one FIB, while Cases 5 and 6 exploit two. In sum, there are three VMs in Case 1, while Case 6 requires 15. Regardless of the case, NGCore is always at the center of the network.

Considering the evaluation methodology, performance tests encompass aspects from NG, FIXP, and the host machine. First, NG presents the round-trip time for completely publishing (RTT_{PUB}) and subscribing (RTT_{SUB}) a single photo to/from NGCore. Meanwhile, RTT overheads compare the relative latency increase from an FIXP scenario to Case 1 (i.e., without FIXP). Nevertheless, these measurements are influenced by the virtual networks and NG processes.

Therefore, FSW processing time (PT_{FIXPSW}) outlines the required time to forward a packet at each FSW. Moreover, NGCA processing time ($PT_{RuleAdd}$) encompasses the time to add a new entry at the FDP and receive its feedback (Fig. 2, steps 9–11). Considering the host machine that virtualizes these network topologies, we analyze the machine stress by assessing the CPU and RAM usage for each network topology and their overhead, taking Case 1 as the reference.

It is crucial to highlight that the proposed evaluation comprises 25 experiments for each scenario. In each case, NG hosts exchange 1000 photos, providing the simple average of RTT_{PUB} and RTT_{SUB} for 1000 photos with a confidence interval of 95 percent. Meanwhile, the PTs derive from the Wireshark network analyzer, which captures the NG packets required to transfer about 100 photos, and a custom Python script helps to obtain the PT_{FIXPSW} and $PT_{RuleAdd}$ from the network log. Meanwhile, a Bash script oversees the host performance.

Results

Table 3 synthesizes the proposed scenarios' results in a Dell PowerEdge 7640 with two processors Intel R Xeon™ Silver 4114 10 Core and 256 GB (8x32GB) RDIMM DDR4 2667 MT/s. In this host, each Ubuntu Linux 16.04 VM has 20 GB of HD, 32 GB of RAM, and three processor cores are connected through VirtualBox's virtualized networks.

The RTT_{PUB} and RTT_{SUB} present the NG performance. As expected, these end-to-end measurements have consistently increased as FDP increases in the number of FIXP instances. The RTT overhead is appropriate for this scenario, once it takes 19.7/15.1 percent at the largest FDP. Nevertheless, NG and VirtualBox's processes impact these results.

In turn, PT_{FIXPSW} and $PT_{RuleAdd}$ relate to the FIXP performance. The first column points out that an FSW takes about 6.5 ms to forward a packet to its destination on average. Interestingly, this is mostly due to the central FSW that experiences a higher throughput to forward data, and BMv2 might restrict its performance. From the tests, its PT_{FIXPSW} is usually more than 5 ms, while the others FSWs present a value around 1 ms. Regarding $PT_{RuleAdd}$, NGCA experiences about 126 ms to add one rule in the FDP and receive its respective FAck. For any case, it presents an average throughput of 200 kb/s in the FDP and 11 kb/s in the FCP. As expected, the management throughput decreases as the underlying FDP is set, but NGHellos are still multicasted to the FCP.

The remaining columns display the host stress over varied virtualized network topologies. Surprisingly, these parameters outline a minimum change (8.9 percent CPU and 8.3 percent RAM), and the most significant difference occurs between Scenarios 1 and 2 (3.6 percent CPU and 2.8 percent RAM).

Complexity and Benchmarking Analysis

The presented methodology has analyzed the interconnection of three NG hosts with varied topologies. The obtained results are promising once FIXP adds an acceptable overhead for the NG and the host machine. Regardless of the network complexity, the impact seems contained. Nevertheless, we present a controlled throughput due to the virtualized setup, which hinders BMv2 performance and can impact the quality of service offered. This throughput is not even close to P4 limits, and P4-ready network switches can improve our findings.

For comparative purposes, it is challenging to compare our performance to the related works.

FIXP infrastructure and NGCA efficiently set FDP on the fly. Moreover, their cooperation has not significantly affected the NG application.

First of all, some works are not similar since they do not exploit VMs or P4 ([11] and [14]). Even though some present P4, they have not disclosed their controllers' development (e.g., a custom Python controller to interact with Stratum [12]). This choice differs from the NGCA, which is scalable to NG and ready with any NG Ethernet scenario. Furthermore, the evaluated throughput in [11, 12] reach up to 1 Gb/s, but this is still a performance aspect to be assessed on FIXP. Concerning the latency, [14] experiences a higher overhead due to the compatibility between multiple architectures. The obtained overhead from [11, 13] is comparable with our solution.

FIXP infrastructure and NGCA efficiently set FDP on the fly. Moreover, their cooperation has not significantly affected the NG application. Considering the $PT_{RuleAdd}$ overhead, it only impacts the required time to set the virtual circuit's forwarding paths, not the content exchanged.

Open Issues and Future Directions

One major open issue lies in the virtualization technique and the automation for conceiving network topologies. Even though CPU and RAM usages have not varied significantly, each VM requires 20 GB of hard disk, and each network connection is manually set. In addition, we cannot present the FIXP delays profile due to the synchronization between VMs.

Future works must consider other performance aspects like scalability, throughput, and packet loss. In such opportunities, it can exploit testbeds, other virtualization techniques, and P4-ready hardware, avoiding the current BMv2 bottleneck.

FIXP future works may extend its scope to relevant aspects, presenting the coexistence of disjointed architectures, network load balancing, and a FIXP application layer for supporting its infrastructure. Considering NG future works, PGCS can consult NRNCS to discover known alternatives for forwarding traffic, allowing new routing techniques to take advantage of published names and contents. In summary, the best configuration of physical infrastructure is directly derived from services' contracts, which could be also immutably stored in a blockchain-as-a-service (BaaS) approach. Artificial intelligence could also be trained and operate over NRNCS data or a peer BaaS. NG can contribute to enable an intelligent and adaptable forwarding plane over heterogeneous NG-SDN.

Conclusion

This article addresses the design of adaptable NovaGenesis (NG) data and control planes for FIXP. This work has synergy with NG-SDNs and NGNs in dynamically managing a programmable network through an environment where multiple architectures can coexist and provide their best service. NG has unique novelties that justify our interest in continuing this project, such as protocol implementation as services, contract-based operation, and semantics-rich self-organization. Our solution supports all NG features, including the ones important for content distribution, such as packet fragmentation and broadcast at the data plane. The obtained results are promising for both our proposal and FIXP as an SDN multi-architecture environment. Even though these results represent some tens of milliseconds, they derive from a virtualized network, which may decrease with physical P4-enabled hardware implementations.

Acknowledgments

This work was supported partially by RNP, with resources from MCTIC, Grant No. 01245.010604/2020-14, under the Brazil 6G project of the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações — CRR) of the National Institute of Telecommunications (Instituto Nacional de Telecomunicações, Inatel), Brazil; by grant #2015/24518-4 from FAPESP through the FIXP project; and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) — Finance Code 001. The authors also thank CNPq and FAPEMIG.

References

[1] T. Qiu *et al.*, "How Can Heterogeneous Internet of Things Build Our Future: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 20, no. 3, 2018, pp. 2011–27.
[2] A. Antonopoulos *et al.*, "Shedding Light on the Internet: Stakeholders and Network Neutrality," *IEEE Commun. Mag.*, vol. 55, no. 7, July 2017, pp. 216–23.
[3] J. Pan, S. Paul, and R. Jain, "A Survey of the Research on Future Internet Architectures," *IEEE Commun. Mag.*, vol. 49, no. 7, July 2011, pp. 26–36.
[4] B. Ahlgren *et al.*, "A Survey of Information-Centric Networking," *IEEE Commun. Mag.*, vol. 50, no. 7, July 2012, pp. 26–36.
[5] J. Day, *Patterns in Network Architecture: A Return to Fundamentals.*
[6] F. de Oliveira Silva *et al.*, "Enabling Future Internet Architecture Research and Experimentation by Using Software Defined Networking," *Proc. Euro. Wksp. SDN*, 2012, pp. 73–78.
[7] D. Kreutz *et al.*, "Software-Defined Networking: A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, 2015, pp. 14–76.
[8] J. Ni, X. Lin, and X. S. Shen, "Efficient and Secure Service-Oriented Authentication Supporting Network Slicing for 5G-Enabled IoT," *IEEE JSAC*, vol. 36, no. 3, 2018, pp. 644–57.
[9] P. Bosshart *et al.*, "P4: Programming Protocol-Independent Packet Processors," *SIGCOMM Comp. Commun. Review*, vol. 44, no. 3, July 2014, pp. 87–95.
[10] A. M. Alberti *et al.*, "Advancing Novagenesis Architecture Towards Future Internet of Things," *IEEE IoT J.*, vol. 6, no. 1, 2019, pp. 215–29.
[11] O. Karrakchou, N. Samaan, and A. Karmouch, "ENDN: An Enhanced NDN Architecture with a P4-Programmabie Data Plane," *Proc. ACM Conf. Information-Centric Net.*, 2020.
[12] S. Gimenez, E. Grasa, and S. Bunch, "A Proof of Concept Implementation of a RINA Interior Router Using P4-Enabled Software Targets," *Proc. Conf. Innovation in Clouds, Internet and Net.*, 2020, pp. 57–62.
[13] W. Feng, X. Tan, and Y. Jin, "Implementing ICN over P4 in HTTP Scenario," *Proc. Intl. Conf. Hot Info-Centric Net.*, 2019, pp. 37–43.
[14] "Exploring Interoperability Assessment for Future Internet Architectures Roll Out," *J. Net. Computer App.*, vol. 136, 2019, pp. 38–56.
[15] J. Gavazza *et al.*, "Future Internet Exchange Point (FIXP): Enabling Future Internet Architectures Interconnection," *Advanced Info. Net. and Appl.*, Springer, 2020, pp. 703–14.

Biographies

Thiago Bueno da Silva holds M.Sc. (2021), B.Sc. (2018), and M.B.A. (2020) degrees from INATEL, Santa Rita do Sapucaí, Brazil, a Technology Management Certificate (2016) from the University of California Santa Barbara, and an Electronics Technician (2012) degree from ETE "FMC," Santa Rita do Sapucaí, Brazil. He has experience in the IT area, researching future Internet, software-defined networks, Industry 4.0, and smart cities.

Fábio Luciano Verdi is an associate professor in the Computer Science Department at Federal University of São Carlos (UFSCar). He has been working on data centers, cloud computing, SDN, and data plane programmability. He is a member of the LERIS Research Group and has been leading projects in the areas of computer network monitoring, virtual resources, and cloud infrastructures.

Juliano Coelho Gonçalves de Melo received a B.Sc. (2003) and an M.Sc. (2019) in computer science from the University of Uberlandia, Brazil; and is currently pursuing a Ph.D. in computer science at the same university. He has more than 10 years of experience in the IT area. His research interests include 5G, IoT, SDN, cloud computing, future Internet architecture, and big data.

José Augusto Suruagy received a Ph.D. in computer science from the University of California Los Angeles in 1990; an M.Sc. in electrical engineering from Universidade de São Paulo (USP), Brazil, in 1982; and a B.S. in electrical engineering from Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 1979. He is currently full professor and researcher at Centro de Informática, UFPE. His research interests include networking architectures, traffic engineering, network measurements, quality of service, performance evaluation, and elastic optical networks.

Flávio de Oliveira Silva [M] is a professor in the Faculty of Computing at the Federal University of Uberlândia (UFU) and received a Ph.D. degree in 2013 from the University of São Paulo. He is a member of ACM and SBC, and he has had several papers published and presented at conferences worldwide. He is a reviewer for several journals and a member of TCPs of several IEEE conferences. Future networks, IoT, network softwarization (SDN and NFV), future intelligent applications and systems, cloud computing, and software-based innovation are among his main current research interests.

Antônio Marcos Alberti is the head of the ICT Lab at the INATEL, Brazil. In 2012, he was a visiting researcher at ETRI, South Korea. He received his M.Sc. and Ph.D. degrees in electrical engineering from Campinas State University, Brazil, in 1998 and 2003, respectively. Since 2008, he has been designing and implementing a future Internet architecture called NovaGenesis. He has published more than 100 papers.