Future Internet Exchange Point (FIXP): Enabling Future Internet Architectures Interconnection

José A. T. Gavazza, Juliano Coelho Melo, Thiago Bueno da Silva, Antonio M. Alberti, Pedro Frosi Rosa, Flávio de Oliveira Silva, Fábio L. Verdi, and José Augusto Suruagy

Abstract The Internet is a crucial infrastructure for the digital era of a fully connected society. However, the design of the Internet's protocols occurred several decades ago based on entirely different assumptions, and this motivated several initiatives to propose the replacement of the TCP/IP protocol stack. Some of these efforts are known as Future Internet Architectures (FIAs), and some examples of these projects are RINA, MobilityFirst, XIA, CCNx, ETArch, and NovaGenesis. Each architecture has its particular purpose and its own set of design goals, but all of them try to advance several aspects related to the current Internet architecture. Considering that these network architectures use disconnected assumptions, their integration would be impossible. Nevertheless, a possible approach would be the coexistence of a set of FIAs or even interconnection with the current Internet architecture. This work presents the architecture of the Future Internet Exchange

José A. T. Gavazza
Federal University of São Carlos, Sorocaba, Brazil, e-mail: gavazza@gmail.com

Juliano Coelho Melo
Federal University of Uberlândia, Uberlândia, Brazil e-mail: julianoco@ufu.br

Thiago Bueno da Silva
Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, Brazil e-mail: thiagobueno@gea.inatel.br

Antonio M. Alberti
Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí, Brazil e-mail: alberti@inatel.br

Pedro Frosi Rosa
Federal University of Uberlândia, Uberlândia, Brazil e-mail: pfrosi@ufu.br,

Flávio de Oliveira Silva
Federal University of Uberlândia, Uberlândia, Brazil e-mail: flavio@ufu.br

Fábio L. Verdi
Federal University of São Carlos, Sorocaba, Brazil e-mail: verdi@ufscar.br

José Augusto Suruagy
Universidade Federal de Pernambuco, Recife, Brazil e-mail: suruagy@cin.ufpe.br

Point (FIXP), a software-defined infrastructure that will contribute to the deployment of future network architectures by leveraging the concept of current Internet Exchange Points (IXPs). Using a P4 switch for the interconnections of the TCP/IP and ETArch architectures, we implemented a FIXP proof of concept. Obtained results are promising for incoming data packets processing times, rule adding, and flow completion times.

1 Introduction

The current Internet architecture employs different communication protocols and its core is the same as four decades ago. The available technologies, computing capabilities and assumptions for the design of these protocols were very different from those we have today.

Some of today's Internet limitations encompass multimedia applications that require network Quality of Service (QoS), seamless mobility across different access networks, security, and multicast support for efficient content delivery. Another concern is the recent rise of the Internet of Things (IoT), which presents scalability challenges.

Several solutions to solve these and other limitations concerning to the TCP/IP stack have been presented. Nevertheless, their deployments are challenging, and raise issues regarding the management of a complex control plane.

Consequently, the need for a new network architecture for the Internet of the Future has appeared since the late nineties, and this subject has caught the attention of the network research community to date. For example, initiatives such as NewArch [1], Future Internet Design (FIND) [2], Future Internet Architecture (FIA) [3], and Future Internet Architecture Next Phase (FIA-NP) [4] indicate that this subject has been in research since the beginning of this century in the United States. In Europe, FP6, FP7, and H2020 framework programs have funded several projects that have focused on new network architectures, and this topic will also be present in the FP9 framework program under the Next Generation Internet (NGI) initiative [5].

The current scenario concerning FIAs shows that there are several network architectures with prototypes in different stages of implementation, different architectures with specific design goals and communication paradigms. Thus, it is not possible to identify a unique network architecture that comes up as an answer and address all the current Internet limitations. In this way, the quest for a single network architecture capable of unifying the disconnected communication requirements, that can support a new Internet, is a possible race. However, this search does not seem to be the best approach.

This work considers that the best approach is to create the conditions for a Future Internet where different network architectures will be interconnected and executed in parallel in the same infrastructure. In this way, a right place for this interconnection to take place is in a Future Internet Exchange Point (FIXP), which takes advantage of the network's *softwarization* tendencies, such as Software Defined

Networking (SDN), Network Functions Virtualization (NFV) and more recently, network programmability.

This article presents the design and implementation of FIXP, which consists of a set of control plane components, as well as a protocol to program data plane P4-capable switches. FIXP is entirely designed from scratch, using the capability of modern network programmability features. In this paper, we present the FIXP components which are agnostic about the Internet architecture it interconnects, which means that it is easy to add interconnection support to new FIAs as long as they appear.

We did an initial evaluation using two Internet architectures: ETArch [8], an FIA representative, and the traditional TCP/IP (which will naturally co-exist for a long time). The initial evaluation shows that the proposal is feasible, extensible, and lightweight, and in line with current in-network solutions.

This work is organized as follows: firstly, Section 2 presents some background and related work. Section 3 highlights the FIXP architecture, its components and their functionalities. Section 4 presents FIXP implementation and evaluation. Finally, Section 5 concludes the paper and gives some future works.

2 Background and related work

The interconnection of different FIAs to enable the development of multi-architecture applications presents several challenges. The first one is to define the approach to interoperate and/or interconnect different FIAs. Machado et al. [11] presented a taxonomy of possible solutions to combine FIAs and proposed a solution for mapping identifiers of other proposals to XIA. Nonetheless, such idea seems to require a profound adaptation of other FIAs to be compatible with XIA. In Guimaraes et al. [9], it is proposed a framework called Future Internet Fusion (FIFu) that aims to unify existing and future network architectures for transparent interoperability while gradually accommodating new networks. In this scope, a FIFu has an "adaptation layer", in which several interoperable entities called Future Internet Exchange Points (FIXP) emulate a connection endpoint and act as a gateway among architectures, ensuring the interoperation among proposals. Through this hardware, it was feasible to evaluate the premise under three different scenarios. These were a web browsing application, having the Named-Data Networking (NDN) and PURSUIT networks accessing IP web pages, a live video streaming, in which video is available through a multicast in the PURSUIT architecture for NDN and IP clients, and, finally, the last is related with on-demand video, in which the same streamed video of the second scenario was made available on-demand and split into multiple segments for IP and NDN clients.

The second challenge is related to the network hardware required to support the FIXP. Some works, such as [10], proposed a Software Defined Internet Exchange (SDX) to interconnect BGP traffic on the Internet. In this work, there are two separate pipelines: (i) one that is called the Policy Compiler, receiving all participant in-

puts, storing routing tables and routing via BGP-suggested routes; and (ii) the Route Server, which emulates a default route server, receiving suggestions from BGP and calculating the best forwarding route. Nonetheless, this work does not consider the interconnection of different FIAs and it focuses only on the interconnection of BGP WAN (TCP/IP) traffic using SDN-capable switches. On the other hand [9], FIFu is based on the concepts of SDN and implemented through two distinct layers, which enabled the interoperability among distinct architectures. In brief, these are the already mentioned "adaptation layer" and the "intelligent layer", in which the control functions for the adaptation layer are implemented, configured, managing and supporting FIXP operations.

The third challenge concerns the architecture of FIXPs and its components. As a consequence, all the components must be defined as well as the integration among them. In addition, software-defined control is required to deal with unknown data units according to each FIA and to properly configure new flows in the FIXP switch. Moreover, this software-defined control can combine trends such as artificial intelligence, machine learning or deep learning to enable smarter decisions while managing a network topology through the SDN concept, which splits the data, or northbound, and control, or southbound, planes [12].

In this paper, we propose FIXP and its components, as well as the hardware developed to support the approach for interconnecting FIAs. Other challenges should be faced as the FIXP evolves and is deployed.

3 Future Internet Exchange Point (FIXP)

FIXP aims to foster the deployment of new network architectures by enabling the interconnection of physically separated networks, not only based on TCP/IP architecture, but also in alternative architectures, allowing traffic exchanging among them. The concepts of Software-Defined Networking (SDN), Network Functions Virtualization (NFV) and Cloud Computing are the basis for the design of a FIXP. While the Software Defined Exchange (SDX) [10] uses the notion of a software-based infrastructure for Internet traffic exchange, FIXP uses this notion to provide traffic exchange among different FIAs deployed on various ISPs.

The ability to interconnect FIA domains creates the condition to do a large-scale deployment of different proposals, allowing their use for various applications, focusing on what each architecture better offers to them. Interconnection is the crucial feature that enabled Internet dawn and success, and considering that FIXP supports the TCP/IP interconnection, its concept leverages existing IXPs.

The FIXP is the physical location where client Internet Service Providers (ISPs) can connect to one or more provider ISPs, thus creating a network that can span over the globe.

Figure 1 presents an overview of the FIXP concept. In this case, FIXP interconnects different administrative domains of different network architectures, such as TCP/IP and other FIAs. Regarding TCP/IP, FIXP is comparable to the IXP func-

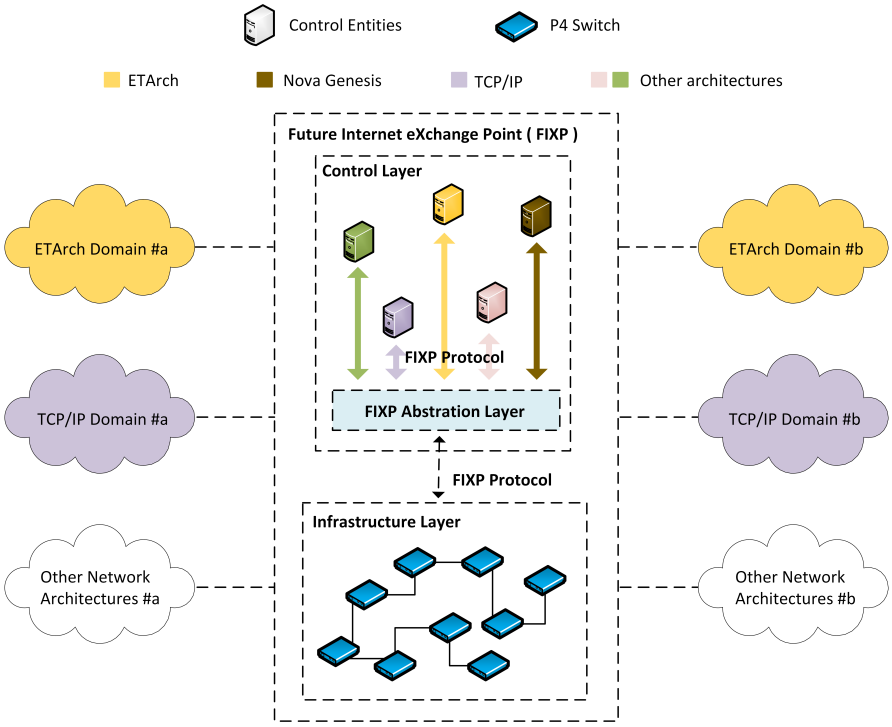


Fig. 1: FIXP architecture overview.

tionality, and the traffic exchange happens by the interconnection of different routers using the FIXP Infrastructure Layer that contains a fabric of P4 capable switches. In this sense, FIXP is capable of exchanging traffic between two or more independent Autonomous Systems, based on the TCP/IP architecture.

The interconnection, as seen in Figure 1, assumes that different domains have a deployment of the same FIA. FIXP is responsible for receiving the packets from that network, for identifying its architecture and for forwarding the data units to the corresponding output FIA, according to its forwarding rules. The Control Layer of the FIXP stores the control entities of each supported FIA that implements its forwarding rules.

The FIXP protocol is responsible for the communication between the Infrastructure and Control layers. The Infrastructure layer receives the data and, if it contains primitives of the control protocols of a supported network architecture, the FIXP protocol encapsulates the primitive and forwards it to the Control layer. The FIXP Abstraction Layer will inspect the control primitive and forward it to its corresponding control entity.

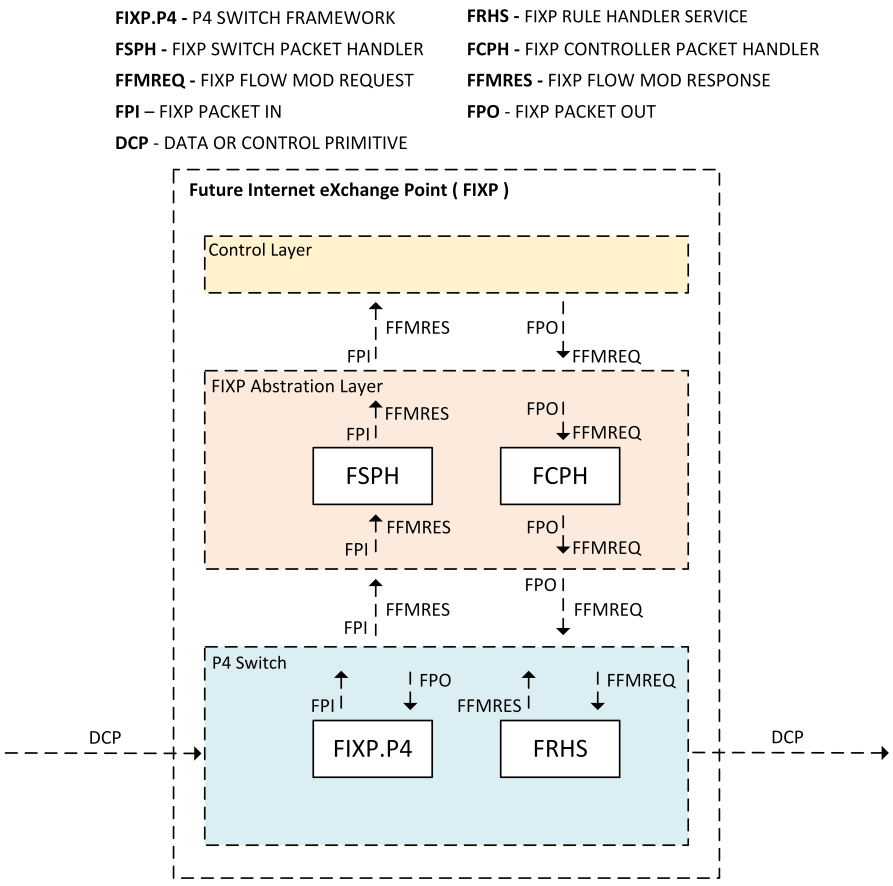


Fig. 2: Internal view of FIXP architecture.

3.1 FIXP Architecture

The subsection presents the FIXP architecture, its components and the primitives of the FIXP protocol responsible for the internal communication.

Figure 2 details the FIXP architecture presenting its internal components. FIXP encompasses a physical switching infrastructure, and, on top of that, a set of virtualized network functions of each FIA. These functions are responsible for controlling the FIXP switches taking into account the integration among different FIAs. A FIXP Switch should be capable of switching different protocol data units (PDUs) used by each network architecture. Thus, it should be capable of handling the different PDUs at line rates. The set of software network functions supports the control plane of each network architecture implemented by FIXP. These components are responsible for

managing and controlling the interconnection rules between two or more different domains. To simplify, Figure 2 presents only one physical switch.

The main modules of the FIXP architecture are the following:

- *FIXP.P4* - This module contains the P4 language implementation that controls the behavior of the switches. To support a new FIA, it is necessary to update the code of this module.
- *FIXP Rule Handle Service* (*FRHS*) - This module runs inside the P4 switch and modifies its internal tables. After this event, it sends a response to notify the status of this modification.
- *FIXP Switch Packet Handler* (*FSPH*) - Handles the FIXP protocol primitives that the P4 switch sends to the FIXP Abstraction Layer. The module inspects the primitive data and forwards it to the corresponding network architecture control entity in the Control Layer.
- *FIXP Controller Packet Handler* (*FCPH*) - This module is responsible for handling the communication of the Control Layer with the infrastructure layer. This module receives all the primitives sent by the Control Layer and forwards them to the corresponding P4 switches of the Infrastructure Layer. In this sense, it hides from network architecture the physical topology of the infrastructure layer.

The communication between these components uses the FIXP protocol primitives. For simplicity, we will only describe their vocabulary and their service without presenting their format. Below we define these primitives:

- *FIXP-PACKET-IN* (*FPI*) - This primitive contains in its payload the data received by the P4 switch in its ingress port. It also contains p4 switches metadata necessary for the proper handling by the upper layers, such as ingress port.
- *FIXP-PACKET-OUT* (*FPO*) - The payload of this primitive is a response that a network architecture needs to send to the infrastructure layer after the occurrence of a *FPI* event. It contains the egress port and the p4 switch where this response should be forwarded.
- *FIXP-FLOW-MOD-REQUEST* (*FFMREQ*) - This primitive contains the data necessary to update the match tables of the P4 switches such as table names, match keys related to the network architecture that requested the operation.
- *FIXP-FLOW-MOD-RESPONSE* (*FFMRES*) - This primitive contains information about the status of a *FFMREQ* such as success or failure.

When a PDU arrives, the *FIXP.P4* module parses the data and identifies which is the corresponding architecture. Every architecture has its match tables, and if no match is found, then a *FPI*) event occurs. The *FPI*) encapsulates this PDU. The switch sends the *FPI* to the *FSPH*.

The *FSPH* forwards the*FPI* to the corresponding control entity in the FIXP Control Layer. In the control layer, the right control entity, associated with the network architecture, receives the *FPI* and provides the behavior associated with that architecture. In an SDN based architecture, the control entity would be a *Controller*, and in the TCP/IP architecture, it would be a *Router*.

The response of a *FPI* is an *FPO* primitive. The control entity encapsulates the response in an *FPO* and forwards the FPO to the *FCPH*. The *FCPH* finally forwards the *FPO* to the P4 switches. The Control layer generates a *FPO* to each *FPI* received.

The Control Layer can also generate a *FFMREQ*. In this case, the control layer forwards the *FFMREQ* to the *FCPH* in order to modify the P4 switches behavior. The *FCPH* finally forwards the *FPO* to the P4 switches. When this happens, the *FRHS* generates a *FFMRES* that is received by the *FPSH* which finally forwards it to the corresponding network architecture control entity in the Control Layer.

After this operation, the P4 switch is ready to handle data plane communication, and, in this case, the switch only forwards the data between ingress ports and egress ports according to the forwarding rules of the switches in the Infrastructure layer.

Considering that these primitives belong to different network architectures, one design goal of the FIXP protocol was that it should work independently of the network architecture. To add support to new network architecture, the only module that needs an update is the *FIXP.P4* to configure the architecture-related information such as match tables and headers. Also, it is necessary to add to the Control Layer the new control entity of this new architecture.

4 FIXP implementation and Proof-of-Concept evaluation

As seen in Figure 2, the FIXP implementation consists of several elements, including a set of switches, an abstraction layer and the controllers. Inside every FIXP switch there are two main applications being executed: the P4 switch code (*FIXP.P4)* and the *FRHS* (presented in Subsection 3.1).

The SDN switch was developed using the P4 packet processing language. It has three main functional sections and follows the traditional P4 pipeline flow: a parser, which receives the packets, extracts Ethernet frames and other protocols and performs the architecture identification of the received packet by analyzing the *ethertype* field from the Ethernet protocol; an ingress section, which defines the packet forwarding tables with the key fields used in the routing rules and the implementation of the forwarding actions for each of the defined FIAs; and, finally, a deparser section, which reassembles the protocol fields on the packets to forward to the correct switch port.

FRHS is an application implemented in Python using the Scapy library, which listens the network interface port connected to the FIXP Abstraction Layer (FAL) and waits to receive control packets. As soon as a packet is received, the *ethertype* field is validated and the data required to create the forwarding rule is extracted from the control packet. The retrieved information is converted into a P4Runtime [6] command and executed. After the command execution, the application uses Scapy to create a response control packet containing the result of the operation and sends it to the controller.

The FAL is composed by two applications: FSPH and FCPH, both implemented in Python using Scapy library. The FSPH uses Scapy library to listen the network interface port connected to the switch. When a packet arrives coming from the switch, the *ethertype* field is verified to identify the network architecture and the packet is redirected to the correct controller through the interface port. The FCPH uses Scapy library to listen all network interface ports connected to the controllers. When a packet is received from the controllers, the *ethertype* field is validated and the packet is redirected to the network interface port connected to the switch.

In the current implementation, there are two controllers implemented, both in Python: an ETArch controller and an IP controller. The IP Controller listen to the network interface port connected to the FIXP Abstraction Layer. When a packet comes to the controller, the *ethertype* field is used to identify the Internet architecture. The controller assembles the packet and sends a control packet to the network interface port connected to the FAL.

The ETArch controller implements the same features described for the IP controller, as well as some of the architecture-specific features, such as registering and unsubscribing hosts to a workspace.

The proof of concept evaluation was conducted in a virtualized scenario where each host, controller, the abstraction layer and the FIXP switch[1] ran on a virtual machine. The benefit of running the FIXP switch over a virtualized environment is the possibility of using the BMv2 to execute the switch.

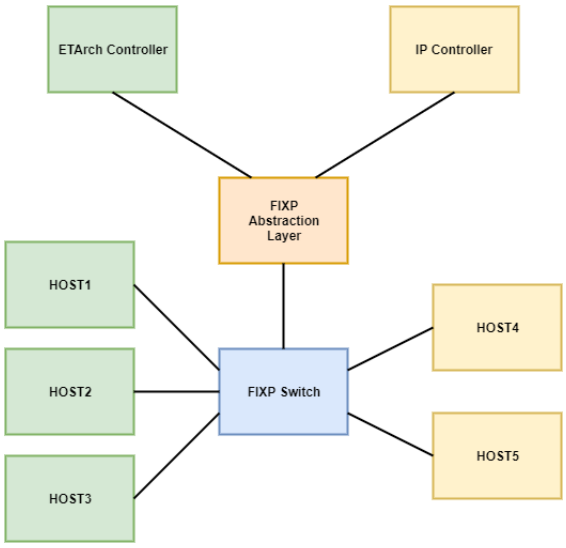
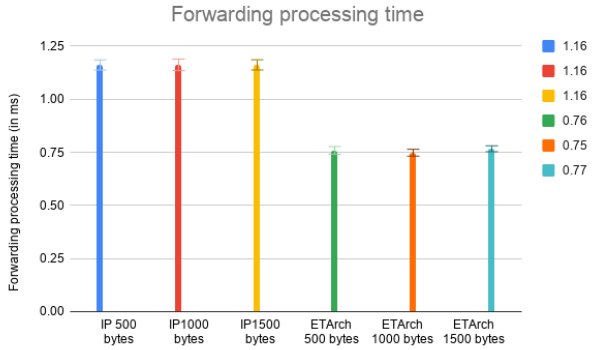


Fig. 3: Topology used on the FIXP evaluation.

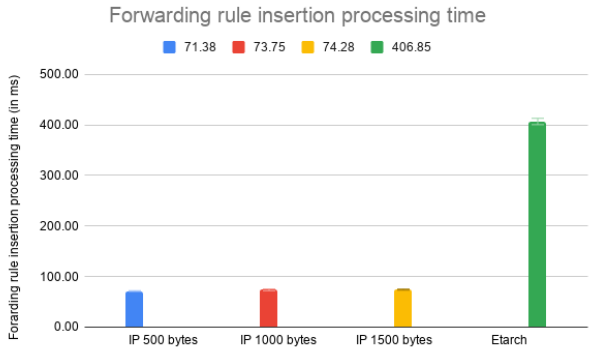
[1] In this initial evaluation, we have only one P4 switch. However, the architecture is generic and can support any number of switches.

In this environment, five hosts have been configured as presented in Fig.3: three of these hosts for ETArch architecture and two hosts for IPv4. One FIXP switch was used, initially with no previous forwarding rules.

Preliminary integration and evaluation tests to verify the behavior of IP and ETarch packets through the FIXP environment were performed. Next, we show the results obtained in the tests. To collect the numbers of Fig. 4, we repeated the test 30 times and 100 packets were sent, varying the size of the packets from 500 bytes up to 1000 and 1500 bytes.



(a) packet forwarding processing time.



(b) forwarding rule insertion processing time.

Fig. 4: Average time to forward data packets and average time to process a forwarding insertion rule.

Fig.4(a) shows the average time to process data packets forwarded by the FIXP switch. Observing the results, we can see that ETArch performed better than the

traditional IP[2]. Moreover, it can be concluded that the packet size does not influence the data packet processing time.

Fig.4(b) presents the average time for adding forwarding rules in the FIXP switch. This time considers the whole processing from the moment that a packet is received in the switch (and there is no match-action) until the rule is installed in the switch. This time considers the full path from the FIXP switch through the FAL and then back to the switch. Note that the time taken by the controllers to decide what to do with the packet is not considered in this whole processing time. Such time is controller-dependent and is not part of the FIXP control plane. In Fig.4b, it can be seen that ETArch has only one measurement for the control plane, which means that the evaluation did not vary with the size of the packets. This was done because the architecture has a standard control packet and the size of the payload does not affect the processing time.

It can also be observed that the average time of control packets processed by ETarch is greater than the time of packets processed by IP. This is due to the fact that ETArch architecture sends three control packets to perform the registration of the entity to a workspace and then send the control packet to the FIXP switch to insert a new forwarding rule.

The last test performed was the flow completion time (FCT). For this evaluation, 2000 packets of 1500 bytes were sent for each architecture. The mean time values obtained in this test are similar to the values presented in Fig.4 for each architecture. For IP, an average of 1.23 ms with a margin of error of 0.16; and for ETArch, we obtained an average of 0.75 ms with a margin of error of 0.81. The value of the margin of error obtained by ETArch results from the large execution time of the control packets of this architecture.

5 Conclusion

In this work, we present the Future Internet Exchange Point (FIXP), its architecture, implementation, and a proof of concept. FIXP aims at interconnecting distinct domains of different Future Internet Architecture (FIA) so that they can co-exist. FIXP is composed of a physical switching infrastructure and a set of virtualized network functions. The physical switching needs to identify PDUs coming from different FIA domains, process them, and forward to the appropriate output port. At the same time, the virtualized functions deal with the control and management of software-defined flows. FIXP allows the development of per architecture software-controllers, decoupling data and control planes for several Internet architectures.

This first proof-of-concept implementation adopted TCP/IP and Etarch as the Internet Architectures showing the capability of FIXP to deal with two different headers. We believe that the TCP/IP architecture will co-exist for a long time, but

[2] Important to highlight that it is not our intention to make comparisons between the Internet architectures. Our goal is to verify the FIXP switch performance for different types of headers and packet size.

yet novel FIAs will appear. Using a virtual P4 switch (BMv2) to interconnect hosts running the TCP/IP stack and ETArch FIA, we evaluated the FIXP. Obtained results are promising for incoming data packets processing times, rule adding on BMv2 switch, and flow completion times. Future work includes an extension for NovaGenesis [7] and NDN FIAs [13], evaluation on hardware P4-based switches, larger-scale experiments, and evaluation for multiple architecture applications.

Acknowledgements We want to thank São Paulo Research Foundation (FAPESP), grant number #2015/24518-4, for funding this project. This study was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

1. NewArch Project: Future-Generation Internet Architecture. https://www.isi.edu/newarch/, 2003.
2. NSF NeTS FIND Initiative. http://www.nets-find.net/, 2009.
3. NSF FUTURE INTERNET ARCHITECTURE PROJECT. http://www.nets-fia.net/, 2010.
4. MobilityFirst Future Internet Architecture Project. http://mobilityfirst.winlab.rutgers.edu/, 2014.
5. Next Generation Internet initiative. https://ec.europa.eu/digital-single-market/en/policies/next-generation-internet, 2019.
6. P4 Runtime. https://p4.org/p4-runtime/, 2019. [Online; accessed 09-December-2019].
7. Antonio Marcos Alberti, Marco Aurelio Favoreto Casaroli, Dhananjay Singh, and Rodrigo da Rosa Righi. Naming and name resolution in the future internet: Introducing the novagenesis approach. *Future Generation Computer Systems*, 67:163 – 179, 2017.
8. F. de Oliveira Silva, M.A. Goncalves, J.H. de Souza Pereira, R. Pasquini, P.F. Rosa, and S.T. Kofuji. On the analysis of multicast traffic over the Entity Title Architecture. In *2012 18th IEEE International Conference on Networks (ICON)*, pages 30–35, 2012.
9. Carlos Guimarães, José Quevedo, Rui Ferreira, Daniel Corujo, and Rui L. Aguiar. Exploring interoperability assessment for future internet architectures roll out. *Journal of Network and Computer Applications*, 136:38 – 56, 2019.
10. Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. SDX: A software defined internet exchange. *SIGCOMM Comput. Commun. Rev.*, 44(4):551–562, August 2014.
11. Michel Machado, Cody Doucette, and John W. Linux Byers. XIA: An interoperable meta network architecture to crowdsource the future internet. *ACM/IEEE Symposium on Architectures for networking and communications systems*, pages 147–158, 2015.
12. Zhihong Tian, Shen Su, Wei Shi, Xiaojiang Du, Mohsen Guizani, and Xiang Yu. A data-driven method for future internet route decision modeling. *Future Generation Computer Systems*, 95:212 – 220, 2019.
13. H. Yuan, T. Song, and P. Crowley. Scalable ndn forwarding: Concepts, issues and principles. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9, July 2012.