# INCA: a mechanism for traffic identification and chaining in the data plane

Guilherme Matos
*Department of Computer Science, UFSCar*
Sorocaba, Brazil
guilherme.matos@estudante.ufscar.br

Leandro C. de Almeida
*Department of Computer Science, UFSCar*
Sorocaba, Brazil
leandro.almeida@estudante.ufscar.br

Luis Miguel Contreras
*Telefonica*
Madri, Spain
luismiguel.contrerasmurillo@telefonica.com

Fábio Luciano Verdi
*Department of Computer Science, UFSCar*
Sorocaba, Brazil
verdi@ufscar.br

***Abstract*—Inside the 5GC (5G Core) a myriad of (virtual) functions may be deployed so that different treatment can be given for traffic coming in and out to/from the RAN (Radio Access Network). One of these key functions is Service Function Chaining (SFC) supported by SRv6 state-of-the-art protocol. In this paper, we present an SFC P4-based solution for traffic identification and chaining using SRv6. We call this function INCA (In-Network Classification and chAining) which is deployed entirely in the data plane using a Netronome Agilo SmartNIC. The INCA is deployed just before the UPF (User Plane Function) inside the 5GC and is capable of observing the traffic coming from and going to the RAN so that it classifies and creates the proper sequence of services to be followed by every specific flow. Our results show that INCA performs the task of packet classification and chaining perfectly with a minimal FCT (Flow Completion Time) impact when compared to the same environment without it.**

***Index Terms*—5G, Service Function Chaining, P4, SRv6**

## I. Introduction

5G was designed with very important structural changes already in place adding the most up-to-date state-of-the-art concepts such as network slicing, SBA (Service-Based Architecture) and also SDN (Software Defined Network). The latter, known as CUPS (Control and User Plane Separation) within the 5G architecture, allows for the separation of the control and user plane which is a key concept since the origin of OpenFlow in 2008. Such a set of technologies and paradigms make it easier to instantiate network functions at the user plane, such as firewalls, cache servers, DPI (Deep Packet Inspection) and others, in order to provide value-added services to the users.

Aligned with the technologies mentioned above, currently we also have witnessed a tendency of adopting in-network solutions leveraging the programmable hardware that has emerged in recent years as well as the usage of network programming languages that enable the creation of new protocols and behaviours inside the network equipment.

Taking into account that we can use network software technologies such as SDN and NFV (Network Functions Virtualization) to create solutions with a high level of programmability, flexibility and modularity, we designed and implemented a solution that uses the SRv6 protocol [1] to solve the lack of a mechanism capable of chaining service functions within the 5GC entirely in the data plane.

INCA (In-Network Classification and chAining) is a P4-based solution capable of performing traffic identification and building an SRv6 header containing an SRH (Segment Routing Header). This in turn contains a list of IPv6 addresses (called segments) representing the functions that are being chained together. To perform this identification, INCA is able to analyze several fields of the protocol stack, such as IPv6 headers (inner/outer), TEID (Tunel Endpoint ID), QoS ID, among others.

In a previous work [2], a proof of concept was accomplished. Using a SmartNIC P4-capable Netronome Agilio CX 2x10Gbe, we carried out the implementation of INCA in a emulated 5G environment, where INCA was able to perform the chaining of functions. Now, in this work, we perform the performance evaluation taking into account the FCT (Flow Completion Time). We performed several tests by sending diferente traffic flows to two different environments, one with and the other without INCA so that we could compare the completion time of each flow for each environment. Our tests show that the use of INCA makes it possible to chain functions efficiently. As for performance, with the use of INCA there was a maximum increase in the FCT of only 1%, which shows that its use has a minimal impact on the 5G architecture.

We organized the remainder of this article as follows. In Section II we present some important concepts for the rest of this work. In Section III, we present some related works found in the literature. In Section IV, we present how our proposal was designed and implemented. In Section V, we show the deployment and evaluation. Finally, in Section VI, we present the final considerations.

## II. Fundamental concepts

This sections presents the fundamental concepts about 5GC and its infrastructure, SFC (Service function Chaining) and SRv6.

### A. 5G core and infrastructure

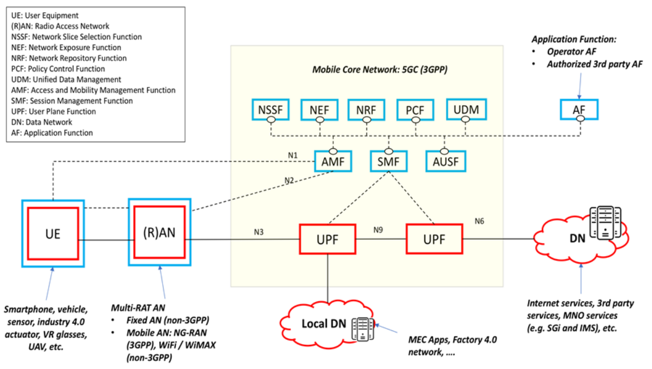
UE: User Equipment
(R)AN: Radio Access Network
NSSF: Network Slice Selection Function
NEF: Network Exposure Function
NRF: Network Repository Function
PCF: Policy Control Function
UDM: Unified Data Management
AMF: Access and Mobility Management Function
SMF: Session Management Function
UPF: User Plane Function
DN: Data Network
AF: Application Function
Mobile Core Network: 5GC (3GPP)
Application Function:
• Operator AF
• Authorized 3rd party AF
NSSF
NEF
NRF
PCF
UDM
AF
AMF
SMF
AUSF
N1
N2
UE
(R)AN
N3
UPF
N9
UPF
N6
DN
Local DN
Smartphone, vehicle, sensor, industry 4.0 actuator, VR glasses, UAV, etc.
Multi-RAT AN
• Fixed AN (non-3GPP)
• Mobile AN: NG-RAN (3GPP), WiFi / WiMAX (non-3GPP)
MEC Apps, Factory 4.0 network, ....
Internet services, 3rd party services, MNO services (e.g. SGi and IMS), etc.

Fig. 1: 5G infrastructure.

Figure 1 shows the 5G components with the blue boxes being the control plane functions and the red boxes the user plane functions. Control plane is responsible for signaling messages and management related to access, mobility, session, policy and other functions. The user plane is where the user data goes through and contains the UE (User equipment), RAN and UPF (User Plane Function). UPF acts as an interconnection point between the mobile infrastructure and the Data Network (DN) and is responsible for routing and forwarding packets between them. This logical network between UE and DN, providing PDU connectivity service, is called PDU Session.

There is a specific protocol stack in the 5GC. Every time a user packet arrives at the RAN, new IPv6, UDP and GTP (GPRS Tunneling Protocol) headers are created, in that order. All these headers will only exist between RAN and UPF. IPv6 and UDP are called IPv6 Outer and UDP Outer, respectively. The UDP Outer header has the port 2152 as destination, and works as an indicator that the data is encapsulated via GTP-U (GTP User Plane Tunneling). The GTP is the most common used protocol for 5G mobile networks and is adopted for data transfers, session management and QoS (Quality of Service). this protocol can be materialized into two different forms: GTP-U for the user plane and GTP-C (GTP Control) for the control plane. A new extension of this header was also specified called PDU Session User Plane Protocol [3], which carries information about QoS and QFI (QoS Flow Identifier). UPF and RAN are the components responsible for creating and dropping the protocol stack. In the uplink, the RAN creates the headers and UPF removes all these additional headers before sending the original packet to the DN. The same happens on the downlink: UPF creates additional headers and RAN drops before sending the original packet to the UE.

5GC offers the possibility to deploy functions on the user plane, but we still need a mechanism to chain service functions within this architecture [4].

### B. SRv6 - Segment Routing over IPV6

SRv6 is the IPv6 data-plane instantiation of Segment Routing [1]. As we can see in Figure 2, it is an extension of IPv6 header that is created with a segment list (a list of IPv6 addresses, called SIDs - Segment IDentification) and a pointer (SL - Segment Left) to identify which segment is active. Every time that the packet passes through a segment endpoint (SR-capable nodes whose address is in the IPv6 destination address) the pointer decreases, and the new SID of the segment list is copied to the destination address. If the packet passes through a non SRv6-aware device, the forwarding is done normally with the destination address. This way, SRv6 integrates both the application and the underlying transport layer into a single protocol, allowing operators to optimize the network in a simplified manner and removing forwarding state from the network.

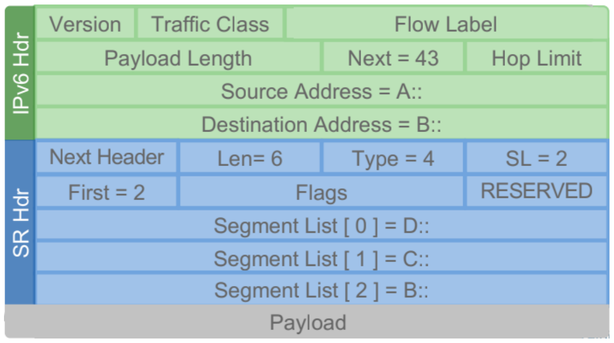

Fig. 2: IPv6 + SRv6 Header.

There are two encapsulation modes using SRv6: encap and inline. The first encapsulates the corresponding packets in a new external IPv6 header containing the SRH. The second mode extends the existing IPv6 and just creates the SRH right after the original packet header. SRv6 is available in the mainstream Linux kernel since version 4.10, but the option to de-encapsulate headers created in inline mode has not yet been implemented [5].

### C. SFC - Service Function Chaining

SFC is a mechanism that allows various functions to be connected to each other in a way that allows for better utilization and integration with SDN. An example of SFC usage is an ASP (Application Service Provider), where there are three VFs (virtual functions) that can be used to provide services to its users. In this case, there is the possibility of having different traffic going through different functions, making it difficult to carry out the chain of these functions. With the use of SFC, we can dynamically implement and interconnect these functions.

## III. Related Works

The work [6] uses SFC to manage the slice life cycle through the adoption of the ETSI NFV Management and Orchestration (MANO) framework, with SRv6 being used to achieve this goal. Unlike this work, we use SRv6 to achieve the service function chain technique to forward the packet to other network functions, which adds value to the service, leaving the creation and management of slices under the responsibility of the existing 5G system. In addition, we use the P4 language to create a programmable device, which was not used in [6].

The work [7] uses P4 to build programmable switches that are located between the backhaul (access network) and the SGW-U (which in 4G networks acts similarly to UPF). These switches are capable of offloading GTP functions, that is, being able to carry out the entire process of encapsulating and de-encapsulating packets in GTP tunnels, thus relieving the VMs' CPUs. Our work is different because it is focused on building an SRv6 header extending IPv6, and with that, enabling the service functions.

In [8], an architecture to both, manage SFC via OpenFlow and use SR-I/OV (Single-Root Input/Output Virtualization) in SDN environments to speed up packet processing, was proposed. VLAN-based rules were used to control forwarding, and that is why it is necessary to install a software client in each service in the chain.. Furthermore, a communication channel was built between the OpenvSwitch and the L2 switch in the SR-I/OV NIC. In comparison, our work already uses a SmartNIC with SR-I/OV enabled. However, instead of creating the SFC management via OpenFlow, we do this via P4 code, which is loaded inside the card. After the creation of the SRv6 header by INCA, each service in the chain needs to know and know how to deal with this header so that the routing between functions occurs correctly. And as this feature is already part of the linux kernel, we avoid installing any client, thus facilitating the deployment of SFC.

## IV. INCA: Design and implementation

According to key aspects of 5G, certain functions may or may not be allocated to customers, and what determines this is the type of traffic flows and other needs related to the requested service. Our solution is intended to be deployed within the 5GC, so that it can be allocated on demand, that is, only if there is a need to perform SFC. To do so, it needs to have a certain level of autonomy in a way that the entire SRv6 creation, dropping and routing to functions do not depend on or impact the underlying functions. Therefore, SRv6 header is a key candidate for keeping the transparency respecting the remaining TCP/IP stack.

A fundamental point is that SRv6 cannot be used as 5G user plane transport protocol since other functions still depend on the GTP, which is the most common protocol used from RAN to UPF communication. As such, INCA is in charge of creating and removing the SRv6 header before forwarding the original GTP traffic to the UPF. The SRv6 is created, the segment list is added and the traffic flow is forwarded to the VFs in a totally independent and transparent way. By doing this before the UPF, we enable SFC inside the 5GC without the need to change the original transport protocol or other predefined functionalities.

To support the aforementioned mechanism, we chose the inline SRv6 mode, as it offers the advantage of using the IPv6 outer header, that already exists in the 5G core, thus avoiding the creation of another IPv6 header. Although the inline mode is more interesting within the 5GC by extending the SRH from the existing IPv6 header, avoiding the creation of a new one, this mode also imposes a challenge. As a natural choice, Linux OS is used to host the VFs acting as both, the SRv6-capable router and the host for the VF. By default, the last function is always responsible for removing the SRH. However, the current Linux Kernel supports only the encap mode, that is, when the SRH is removed, the IPv6 is removed as well, which would cause problems within the 5GC. We solved this problem by encoding the inline mode in P4 and leaving it under the responsibility of INCA, so that it is now possible to decap the SRv6 header without changing the original IPv6 header.

INCA was totally implemented using the P4 language. We added all the headers used in the 5G architecture and also the SRv6. We configured the parser so that INCA accepts packets in the 5G standard, with or without the SRv6 header. Figure 3 shows the INCA processing flow, which indicates the actions to be taken: add the SRH or drop the SRH. The packet models that are supported can be seen in Figure 4, items 2 and 4, which are respectively the default 5GC protocol stack and the default stack with the addition of the SRH. For sake of simplicity, the ethernet headers do not appear in Figure 4. However, it is the first header to be opened by INCA. After the identification of the headers, if it is within the standards mentioned above, the packet is accepted and then enters the ingress processing step, where for each situation there is an action:

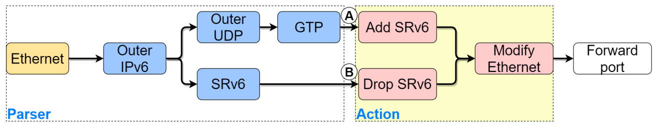

Fig. 3: INCA processing flow.

- Case A - If the packet does not contain an SRH, a search is performed in the tables configured by the control plane to check if there are any rules that apply to this packet. If it exists, the SRH is created and the packet is forwarded to the first network function;
- Case B - If the packet contains an SRH with SL field equals to zero it means that the packet has already gone through all functions, and INCA drops the SRH (inline mode) and forwards the packet to the UPF.

Figure 4 shows how our solution works. After the user (UE) sends a packet (1) and this packet is encapsulated by the RAN (2), the INCA receives it. This scenario fits into Case A: an SRH is created and the packet is sent to the first network function (3). This network function identifies the

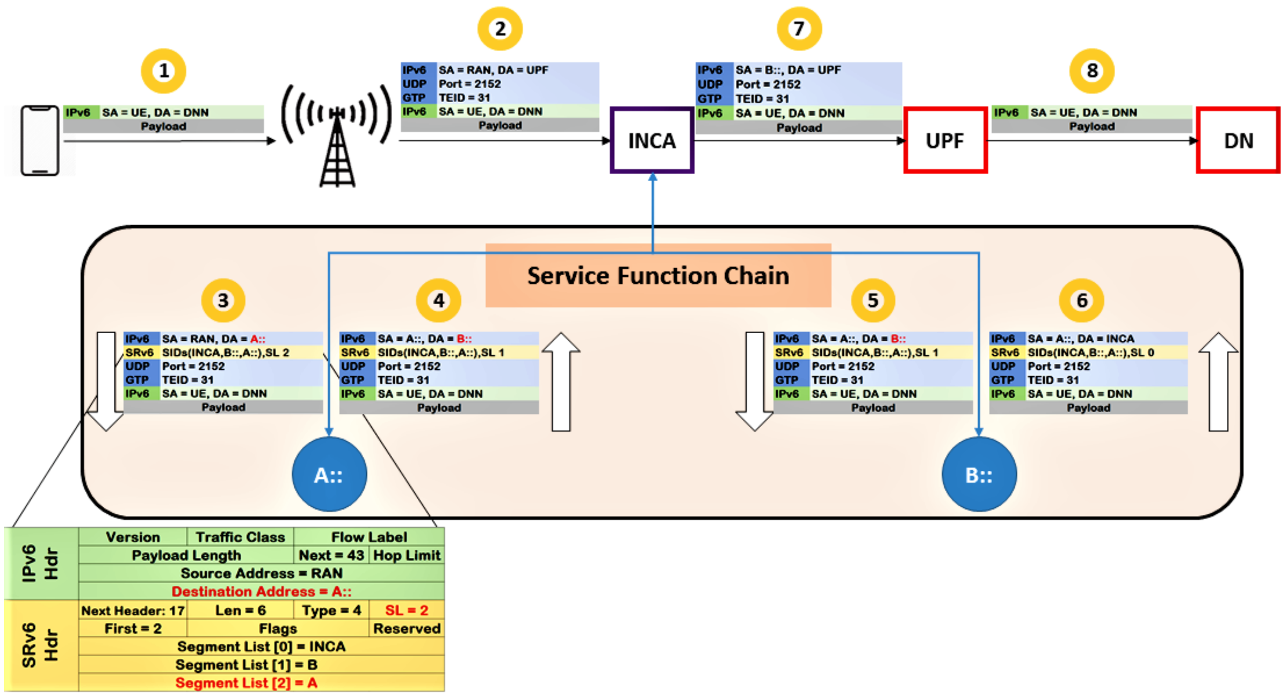

Fig. 4: INCA working flow.

SRH, performs its operation (executes the VF), decrements the SL field, updates the IPv6 destination based on the new SID (4) and sends the packet to the next network function (5). This sequence is then repeated, however, VF B:: is the last in the sequence so the packet is sent back to INCA (6). INCA observes that the SL field is equal to zero (Case B) and that there are no more network functions to go through. So INCA removes the SRH and forwards the packet to the UPF (7) which forwards to the final destination (8). Note that the packet forwarded to the UPF is identical to the packet received by the RAN.

In Case A, several fields from the user transport, network layer and from the 5GC can be used as correspondence by the control plane to identify a traffic. The PDU Session User Plane Protocol [3] was also implemented as one of these fields, which is a new extension for the GTP created specifically for the 5GC. INCA has already implemented all the necessary mechanisms to read this protocol and even perform matching using QoS ID, thus creating SFC based on this value. In this way, INCA is able to perform SFC based on IPv6 addresses, UDP or TCP ports, slice ID, QoS ID, or any combination of these.

## V. Deployment and Evaluation

In this section, we present how we implemented and evaluated INCA. We used a P4-capable Netronome Agilio CX SmartNIC with 2x10Gbps where INCA was deployed. We also used Virtual Machines (VMs) with Linux OS for the 5G environment and to host the VFs.

### A. *Deployment*

Using SR-I/OV, the Netronome card is able to create VFs logically isolated from PFs (physical functions) in order to share the physical resources. Typically, VFs are assigned to VMs (virtual machines). The access to VMs is done directly via PCI, avoiding the kernel host. Our solution uses five virtual interfaces (VF_1 - VF_5) shared from the two existing physical interfaces.

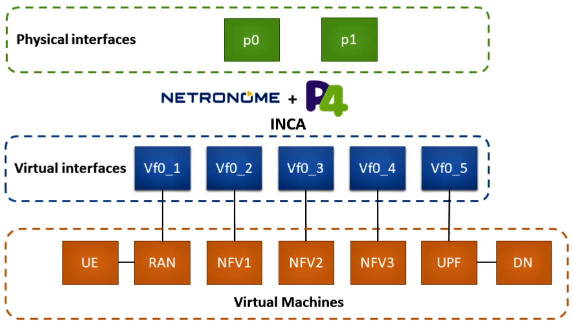

Fig. 5: Testbed setup.

According to Figure 5, the following configuration was defined: seven VMs, three of them are network functions (NFV1 is an IDS - Intrusion Detection System, NFV2 is an IPS - Intrusion Prevention System and NFV3 is a Firewall - Packet Filter). One VM for the UE, which sends traffic to the DN, and another VM for DN, which responds to UE requests. Lastly, we have the native functions of 5GC: RAN and UPF. These last two perform the encapsulation and de-encapsulation of packets in GTP tunnels, and for this, we use

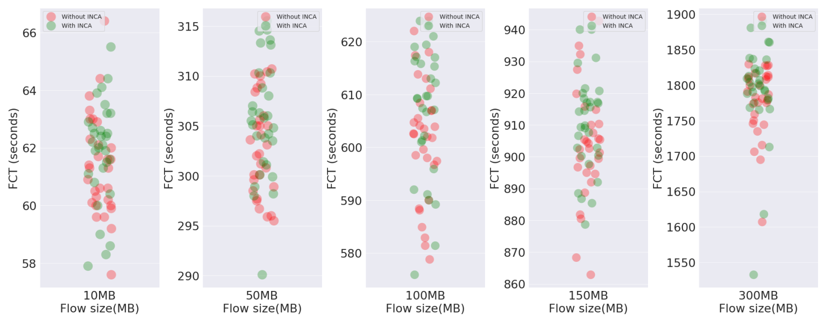

Fig. 6: Time distribution in relation to flow size.

a P4 code deployed in a BMv2 (Behavioral Model version 2) virtual switch running on these two VMs.

The network functions were configured in order to exemplify several options for chaining from different scenarios. Thus, two scenarios were set up. In the first one, the SFC is formed by the network functions NFV1 and NFV2, which perform their functions as IDS and IPS as well as SRH processing. In the second scenario, the SFC is formed by NFV2 and NFV3. The latter is a firewall that is configured to block traffic coming from the UE. Using these two scenarios, we can both send flows and dynamically change the network functions so that a given traffic is allowed while other is blocked by the firewall.

### *B. Evaluation*

In a previous work [2], we showed how INCA works with all its resources. In this work, our objective is to show the tests related to INCA's performance and evaluation.

Our tests are based on the evaluation of INCA's impact on the FCT in relation to the same environment without the use of INCA. We configured INCA so that SRH encapsulation (creation) and de-encapsulation (destroying) actions are done in sequence. In this scenario, packets do not go through network functions. Instead, the packet arrives at INCA where the SRH is created in the ingress processing. Right after the SRH creation process, it is destroyed and then the packet is forwarded to the UPF.

To perform the evaluation, we use Iperf [9] to send TCP flows of different sizes: 10MB, 50MB, 100MB, 150MB and 300MB. The experiments were done 30 times for each flow size in each environment.

For this work, we took into account a reasonable number of three SIDs. The rules and policies on what to do with traffic flows can be configured by the control plane to build SRv6 header with one, two or three SIDs.

Our tests showed that the variation in the number SIDs created has no influence on the FCT. So, we chose the environment where two SIDs are created to perform the comparison with an environment without INCA.

Tables I and II show the summary of the tests performed and Figure 6 shows the distribution and the comparison of the FCT for each flow size, with and without INCA. On the X axis, we have the size of the flows, and on the Y axis we have the completion times for each test in seconds. The dots in green refer to environment with INCA results and red the environment without it.

TABLE I: Summary of tests without INCA

| Flow Size | Min | Max | Mean | Median | Std |
|---|---|---|---|---|---|
| 10MB | 57.6 | 66.4 | 61.37 | 61.3 | 1.77 |
| 50MB | 295.5 | 310.7 | 302.92 | 302.65 | 4.82 |
| 100MB | 578.8 | 622 | 601.9 | 602.55 | 11.07 |
| 150MB | 862.9 | 942.7 | 903.46 | 903.4 | 17.71 |
| 300MB | 1607 | 1829.5 | 1776.49 | 1784.85 | 49.03 |

TABLE II: Summary of tests with INCA

| Flow Size | Min | Max | Mean | Median | Std |
|---|---|---|---|---|---|
| 10MB | 57.9 | 65.5 | 61.88 | 62.15 | 1.8 |
| 50MB | 290.1 | 315.5 | 305.73 | 305.65 | 5.9 |
| 100MB | 575.9 | 623.9 | 607.19 | 609.5 | 12.81 |
| 150MB | 878.7 | 940 | 910.37 | 912 | 15.17 |
| 300MB | 1532.5 | 1890.5 | 1791.89 | 1800.5 | 70.27 |

Analyzing Tables I and II, we can see that the mean, median and standard deviation of INCA were mostly higher when compared to the same environment without INCA, which indicates a higher FCT.

As for the amplitude of the results, that is, the difference between the highest and lowest results of each flow, it is possible to see in Figure 6 that both environments had variations. This demonstrates that the transmission rates did not remain constant during the tests.

Figure 7 shows the FCT with and without INCA. We can visualize the median, dispersion of results, interquartile range and outliers. These graph show that in all cases the median referring to INCA is higher than without INCA. Furthermore, we can note the following patterns: the median of INCA is always close to Q3 (top edge of rectangle) of the environment without INCA and Q1 (botton edge of rectangle) of INCA always starts close to the median of the environment without INCA. This demonstrates that the interquartile ranges (that is, the dispersion of results) of both tests are very close.

In Figure 8, the distribution of values related to the difference between the environment with and without INCA is shown. In the X axis, we can see the distribution of the difference and in the Y axis, the values are in percentage.

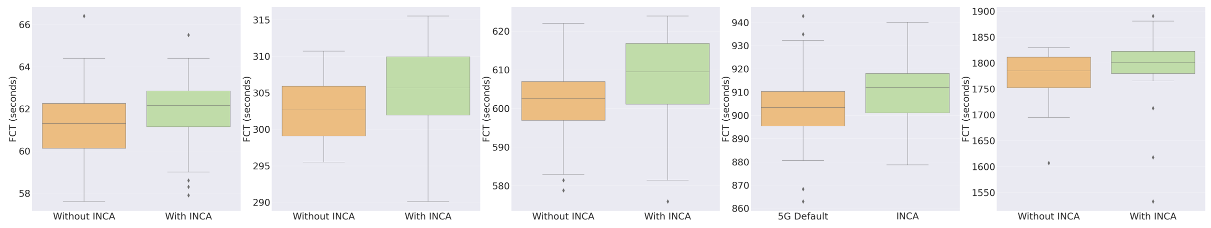

Fig. 7: FCT comparison.

In fact, looking at this figure, it is possible to observe that there is a small increase between 0.77% and 1.01% in the FCT when INCA is used.

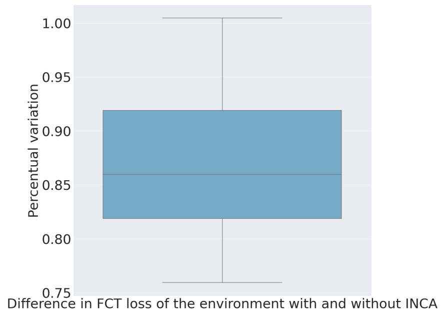

Fig. 8: Percentual variation.

The results show that INCA is able to perform SFC within the 5G architecture effectively. It was also possible to observe that the creation of up to three SIDs does not change the FCT. INCA has a increase of 1% in FCT when compared to the scenario without it. We also noticed that there is a pattern, where FCT values with and without INCA are overridden, so that the results always remain within a pre-defined range where it is possible to predict the behavior, which is an important feature for SLA (Service Level Agreements).

## VI. Conclusion

In this work, we presented the design, implementation and evaluation of INCA, a solution developed in P4 and deployed in a SmartNIC to enable SFC and traffic forwarding within the 5GC.

Our tests show that INCA is able to perform packet identification, building and removing the SRH as well as perform routing between network functions.

Performance tests show that INCA's impact on FCT is minimal when compared to the same environment without INCA. Furthermore, the use of SRv6 without changing the existing GTP-U enables it for unrestricted use in 5G architectures.

Future works include the implementation of INCA using real RAN and UPF functions, in addition to carrying out tests and evaluation of the implementation of INCA in an edge environment.

## References

[1] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," RFC 8402, Jul. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8402.txt

[2] G. Matos, F. L. Verdi, L. M. Contreras, and L. C. de Almeida, "When srv6 meets 5g core: Implementation and deployment of a network service chaining function in smartnics," *2021 P4 Workshop*, 2021.

[3] ETSI, "5G; NG-RAN; PDU session user plane protocol (3GPP TS 38.415 version 16.4.0 Release 16)," European Telecommunications Standards Institute (ETSI), TECHNICAL SPECIFICATION (TS) TS 138 415, 04 2021, version 16.4.0.

[4] S. Homma, X. de Foy, and A. Galis, "Gateway Function for Network Slicing," Internet Engineering Task Force, Internet-Draft draft-homma-nfvrg-slice-gateway-00, Jul. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-homma-nfvrg-slice-gateway-00

[5] D. Lebrun and O. Bonaventure, "Implementing ipv6 segment routing in the linux kernel," in *Proceedings of the Applied Networking Research Workshop*, ser. ANRW '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 35–41. [Online]. Available: https://doi.org/10.1145/3106328.3106329

[6] D. Borsatti, G. Davoli, W. Cerroni, and F. Callegati, "Service function chaining leveraging segment routing for 5g network slicing," in *2019 15th International Conference on Network and Service Management (CNSM)*. IEEE, 2019, pp. 1–6.

[7] C.-A. Shen, D.-Y. Lee, C.-A. Ku, M.-W. Lin, K.-C. Lu, and S.-Y. Tan, "A programmable and fpga-accelerated gtp offloading engine for mobile edge computing in 5g networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 1021–1022.

[8] H.-E. Tseng and S.-H. Shen, "A low latency service function chain with sr-i/ov in software defined networks," *Wireless Networks*, vol. 26, no. 6, pp. 4459–4475, 2020.

[9] A. Tirumala, "Iperf: The tcp/udp bandwidth measurement tool," *http://dast. nlanr. net/Projects/Iperf/*, 1999.