D1.2: Final Description of Micro-Services and Preliminary Architecture

Bertrand Mathieu[1], Stéphane Tuffin[1], Olivier Dugeon[1], Joël Ky[1], Guillaume Doyen[5], Marius Letourneau[2], Hichem Magnouche[2], Philippe Graff[3], Xavier Marchal[3], Thibault Cholez[3], Edgardo Montes de Oca[4], and Huu Nghia Nguyen[4]

[1]Orange
[2]UTT
[5]IMT Atlantique
[3]Loria
[4]Montimage

Abstract

Following the review of low-latency (LL) services and candidate technologies achieved in deliverable D1.1, the present document aims at exposing the functional description of the micro-services selected by the project and shape the general architecture which will host them. As such, this deliverable D1.2 first exposes the general architecture we envision consisting of three topological levels, from the local one up to the national one. The latter is refined with an analysis of the different deployment levels that a node can implement within this global architecture. Indeed, two types of micro-services are considered in the project: (1) those directly focusing on data-plane processing and deployed on P4 hardware devices, and (2) those requiring more resources and deployed on control-plane through lightweight virtualization. Then, the deliverable exhaustively specifies the expected functionalities of the different micro-services we will implement and deploy. These micro-services are related to packet processing, monitoring and security of the architecture. Finally, some prospective use cases, which represent relevant industrial scenarios, involving network function disaggregation are exposed to identify further developments of the project.

Contents

Chapter 1

Introduction

The MOSAICO project aims at designing, implementing and validating solutions which will push forward the development of Low Latency (LL) services. Previously, the project identified the different classes of services with LL constraints, and gave, based on an in-depth study of the related literature, target key performance indicator (KPI) values for offering satisfying user experiences and clustered the services according to the specific causes of network latency affecting them. This guided the current research and development work to focus on two network latency reduction techniques: reducing queuing delays and latency-optimized network function placement. Among all the studied services, the project selected cloud gaming as its core case study since it is an already deployed service for which the operation can be measured, assessed, and where the limits and potential improvement can be proposed, implemented and validated. Besides, this service can be seen as a relevant baseline for subsequent services which will likely occur in a near future (e.g., Metaverse through cloud-VR serices) due to the expected similarities in terms of network traffic types (e.g., real-time player control, real-time multimedia streaming).

Driven by this choice, we explored the different architectural solutions that could host network-related enhancements for an end-to-end cloud gaming service, as well as the micro-services that could provide relevant support for such service delivery. Thus, in this deliverable, we synthesize this specification work which will be made concrete in the other tasks of the project. Namely, Task 2 that concerns the implementation and validation of the micro-services, and Task 3 that will design and validate an appropriate placement and routing solution for the micro-service ecosystem and technologies developed in the project. Finally, Task 4 will host all testbed related activities which will eventually lead to several partial, or if possible, one single complete framework supporting all the project achievements.

In this context, in the following we present the general architecture we envision that includes three topological levels, from the local one up to the national one. The latter is refined with an analysis analysis of the different deployment levels that a node can implement within this global architecture. Indeed, two types of micro-services are considered in the project: (1) those directly focusing on data-plane processing and deployed on P4 hardware devices, and (2) those requiring more resources and deployed on control-plane through lightweight virtualization. Then, the deliverable exhaustively specifies the expected functionalities of the different micro-services we will implement and deploy. These micro-services are related to packet processing, monitoring and security of the architecture. Finally, some prospective use cases, which represent relevant industrial scenarios, involving network function disaggregation are exposed to identify further developments of the project.

This deliverable is organized as follows. Section 2 presents MOSAICO's architecture: globally and at each topological level. Section 3 provides the functional specification of the set of micro-services forming the scope of the project. Section 4 offers a prospective view on the trends in network function disaggregation the project may further explore. Finally, Section 5 provides the lessons learned and some guidelines for future work.

Chapter 2

MOSAICO Network & Node Architecture

In this chapter we describe the architectural design resulting from the choices made in the MOSAICO project. We first motivate the need for a multi-layer network topology that provides a cost-effective solution to disaggregate network functions into micro-services. Then, we refine this work by specifying a generic topology, and its components, to realistically represents that of a telco. Pursuing this in-depth specification, we then explore different node architectures, each option being examined considering the heterogeneous nature of the micro-services that it will host, separating those related to data-plane operation from those performing an advanced processing whith relaxed latency constraints. Finally, we tackle the case of routing issues between micro-services and we propose to leverage a Segment Routing approach.

2.1 Background: Network function disaggregation

Network function disaggregation is an ongoing trend which aims at splitting monolithic network functions into more granular software components such as micro-services. The disaggregation of network functions often allows to re-think the way monolithic network functions were previously deployed. In this movement, two complementary types of benefits are sought: a) the aggregation of network functions components at higher network aggregation levels b) the simplification of network function components that needs to be deployed at more capillary network locations.

The pooling of components at higher aggregation levels allows to achieve higher utilization of the pooled components, which helps in limiting costs by simplifying capillary network components. In addition, the grouping of components at higher network aggregation levels is often an opportunity to introduce more intelligence due to the availability of more processing power at these locations and to the pooling of components controlling other components located at multiple capillary network locations. To optimize network costs, the software components are deployed at the highest network aggregation level allowed by latency and bandwidth requirements. The mutualization of components and their easier development and maintenance can foster innovation.

Network function disaggregation is often combined with the separation between hardware and software. This is made possible by the virtualization and containerisation which ease software lifecycle management of common off-the-shelf (COTS) hardware platforms, such as Intel or ARM-based servers. These servers can be complemented with hardware accelerators such as FPGA-boards, Smart Network Interface Cards or programmable data-planes, such as P4 switches, when the performance of the COTS platform is insufficient.

The control plane / data plane separation can be seen as an early and simple form of network function disaggregation. However, all monolithic network functions cannot be split according to the control plane / data plane separation, particularly when control and data plane functions are too imbricated to design a relevant control interface between the two. This is, for instance, the case of the lowest layers of the base stations in mobile radio access networks. In addition, the control plane / data plane separation is a coarser separation than what is targeted by the disaggregation. In particular, this is the case where the control plane of a monolithic network function is rearchitected into several micro-services rather than only one software component.

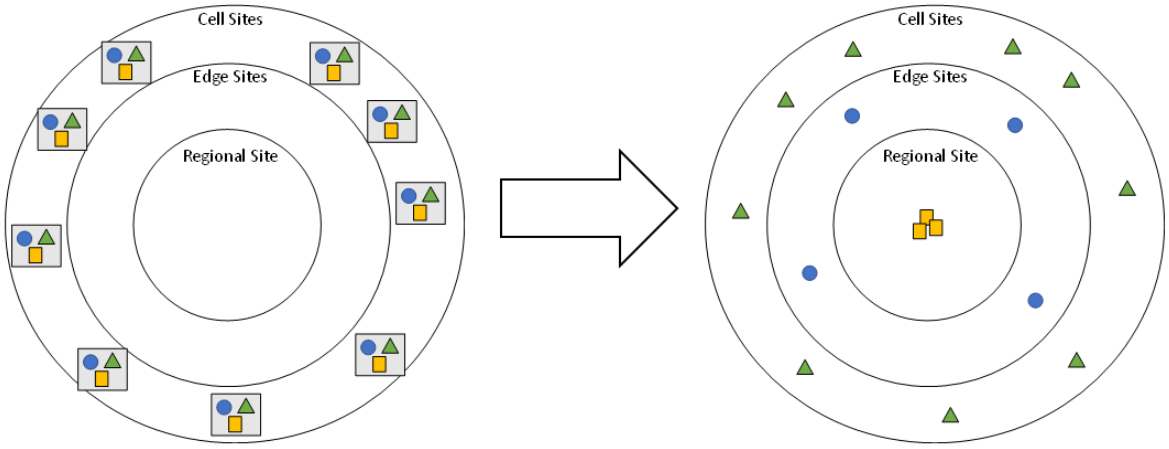

Figure 2.1: Disaggregation and reaggregation of network functions

2.2 Overall architecture

In the MOSAICO project, since we are dealing with low-latency services, we limit the network scope, so as to ensure transmission delays in the order of milliseconds. First, we assume that our targeted service will not be a service hosted in a very far location, but rather on a national or European server. We also assume the core network is over-provisioned. Then, we consider having processing nodes in the access or aggregation part of the network.

Having this in mind, we schematized a generic network infrastructure that: (1) a network operator can implement, and (2) is compliant with the different levels previously identified to potentially support network function disaggregation. We designed a three-level infrastructure: (1) a local part, representing for instance a zone with a diameter of about 50 km; (2) a regional zone, where local zones are deployed, that can have a diameter of about 300 km; and (3) all regional zones are connected with core routers to form the national zone, where peering points enable the connectivity to the worldwide Internet. In Figure 2.2, we picture these three zones in colours orange, green and blue, respectively. It must also be mentioned that we take into consideration both wireline broadband access (e.g. Ethernet or Wi-Fi endpoints connected to an FTTH access through a broadband router) and mobile access (e.g. connecting 5G User Equipements).

In the next paragraph, we detail more precisely each zone and what it can offer.

2.2.1 National zone

The national zone contains the backbone network of an operator. Herein, we can envision having very high-speed routers, transiting data packets from the origin router to the destination router in a very short time. For a country like France, we can have one router per main large region. These routers can be interconnected in a hierarchical and redundant manner for high availability and robustness against failures. Furthermore, this zone can deploy a certain number of routers connected to other Autonomous Systems (AS). As such, it enables any traffic issued by lower zones in the hierarchy to connect to the Internet via peering points. This is represented in blue in Figure 2.2.

The MOSAICO project does not focus its work on this zone because the resources are over-provisioned and, rarely experience latency changes. The internal routing protocols help to reconfigure the network when link failures occur, but queue congestion is uncommon and does not translate into noticeable latency fluctuations in such a zone. Thus, there is little need to further investigate solutions for dealing with latency optimization in this national zone.

2.2.2 Regional zone

The regional zone can be seen as the backhaul network of an operator. It delivers content to users located in a region of about 400,000 km^2. At the points where regional zones are connected to the national zone, a hosting facility provides the infrastructure to host network functions relevant at this topological level. This is what is presented in green in Figure 2.2. Typical network functions located in this part of the

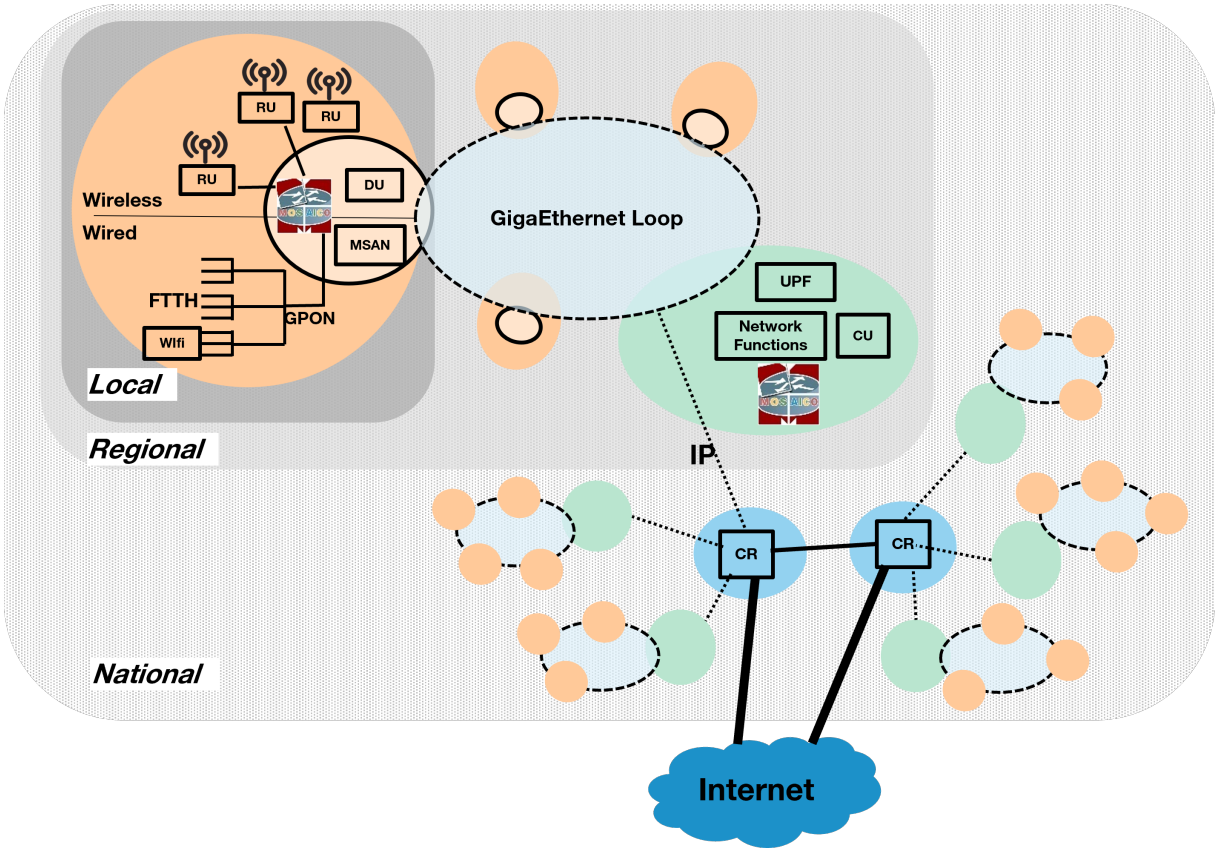

Figure 2.2: The 3-level MOSAICO network architecture

architecture are the User-Plane-Function (UPF) of 5G core networks, the Centralised Unit (CU) the centralized component of disaggregated base stations, or boradband network gateways and functions such as content caching, aimed at optimizing content delivery.

Regarding the micro-services, we will describe in the next chapter some decomposition and deployment strategy of a Cloud Gaming (CG) detector function that can be fully or partly deployed in this zone, among other possible configurations. We can also imagine having monitoring components. One MOSAICO node with a P4 switch and multiple compute nodes can be deployed in such a location for hosting and performing certain functions for assessing and improving the quality and security of the communications.

2.2.3 Local zone

The local zone is the last part of our network architecture and can be seen as the access network (i.e., network edge) of an operator. It corresponds to a small region, like a metropolis or a small region having multiple small towns located close to each other.

This zone is where end-users are connected to the network, either using wired infrastructure such as FTTH or using cellular infrastructure such as 4G or 5G. All connections are managed inside the local center terminating the access links. In this center, there can be some network functions to manage connectivity. In the case of a disaggregated base station, one can find the Distributed Unit (DU). The following network nodes have been identified in the project as queue-forming bottlenecks which could therefore benefit from the AQM and/or transport technologies developed in the project:

- Base Stations of cellular networks: These are subject to queuing delays in the **downlink**. The bottleneck is the radio link which is time-varying.
- Terminals (UEs) of cellular networks: These are are subject to queuing delays in the **uplink**. The bottleneck is the radio link which is time-varying.
- Wi-Fi Access Points: These are are subject to queuing delays in the **downlink**. When the downlink capacity of the wireline interface of the Access Points is larger than the radio link capacity (e.g.,

cases where the Access Point is part of an FTTH Home Gateway). The bottleneck is the radio link which is time-varying.

- Wi-Fi Stations: These are are subject to queuing delays in the **uplink**. When the uplink capacity of the wireline interface of the Station is larger than the radio link capacity (e.g., cases where the Access Point is part of an FTTH Home Gateway). The bottleneck is the radio link which is time-varying.

Consequently, in this zone, we envision deploying our L4S system to ensure the delivery of low-latency services, respecting application constraints. Since this zone should manage less traffic, we can expect the need for a lighter MOSAICO node with a P4 switch and only one compute node deployed in such locations for hosting and performing the required functions.

Overall, we can imagine having several (e.g., # 10) local zones connected to a regional zone and, similarly, a number of regional zones connected to the national nodes.

2.3 Node architecture principles

Following the regional and local zones exposed above, several node configurations can be set up. In this section, we briefly introduce three possible MOSAICO node architectures. Depending on their network location, the MOSAICO node will be different in terms of architecture and capabilities. Since nodes in local zones will be more numerous, for cost efficiency, we should reduce their capacity. We then can imagine having a simple node in a local network, while a more complex node can be deployed in a regional network.

To ensure low-latency delivery of services, we decided to rely on data plane solutions. The promising concept we advocate for data plane network programmability is the Programming Protocol independent Packet Processor language (P4). Some chipsets, like the Intel Tofino [1], are P4-enabled and are already integrated in programmable switches, such as Edgecore switches[2]. Such a device has two levels, the low-level ASIC (Application-Specific Integrated Circuit) for packet processing and forwarding, and the upper level "CPU" for application level processing. The physical links are plugged into the physical ports of the low level, where packets are received/transmitted, and the two levels can communicate via the CPU port.

For control plane functions, we advocate the adoption of an application level programmable environment, leveraging concepts derived from the Network Function Virtualisation (NFV). In this level, micro-services can be containers or processes inside a virtual machine. The hosting environment we selected is Linux due to its wide adoption for these technologies.

2.3.1 Single P4-based hardware switch

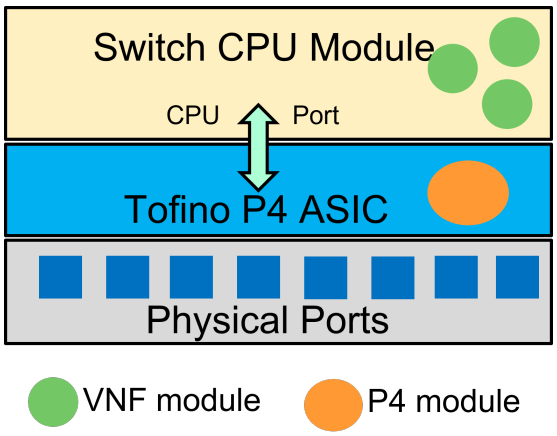

Figure 2.3: MOSAICO Node 1 architecture: P4 Single Switch only

Among the possible architectures we envision, the simpler one is where we deploy only one programmable switch in each local zone (Fig. 2.3). The P4-based functions will manage all the simple

[1] https://www.intel.fr/content/www/fr/fr/products/network-io/programmable-ethernet-switch.html
[2] https://www.edge-core.com/productsInfo.php?id=335

packet processing functions at line rate. However, a switch does not come with only the programmable part (ASIC), but also with a CPU, hosting a Linux environment. This system is dedicated to the configuration of the nodes, but also for hosting the network control plane. For instance, for routing protocols like IS-IS (Intermediate System to Intermediate System), the signaling part will be managed by programs running in the CPU to exchange routing information, compute routing tables, e.g. based on minimum number of hops or, in our case, based on latency, etc.). The latter then communicates with the P4 chipset with the internal CPU port.

After a global review of existing hardware switches, we selected the Edgecore switch, which includes a Tofino P4 chipset and provides good performance and offers technical features useful for us. Indeed, in the CPU, the functions (e.g., the routing protocol control functions) are deployed as containers, one for each function, and the switch provides container management. For the MOSAICO project, we can envision having our VNF functions containerized and deployed to run at the CPU level of the hardware switch.

This solution is the most cost effective one since only one programmable switch needs to be deployed in the local zone. However, we can observe some performance limitations. Indeed, the functions deployed in containers running in the CPU communicate with the ASIC via an internal CPU port, which is not optimized for high bit rates. Thus, if the goal of the functions in the CPU is basically to control some functions deployed in the P4 switch, or processing only a few packets, this solution can be used. But if the VNF deployed to run in the CPU needs to perform packet processing for all or many of the packets transiting via this switch, this solution will fall short.

2.3.2 One P4-based hardware switch and one local compute node

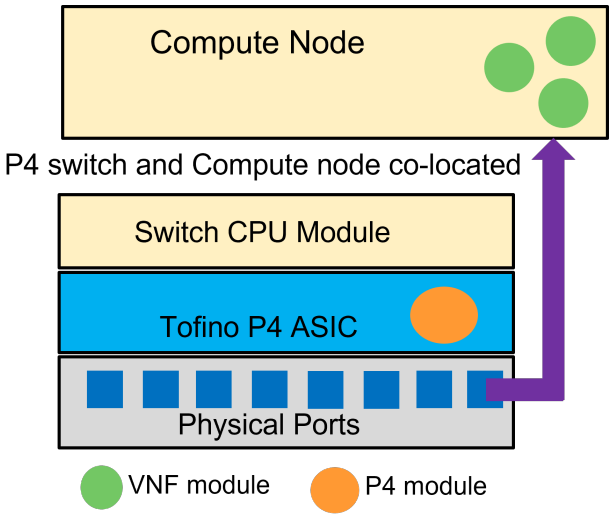

Figure 2.4: MOSAICO Node 2 architecture: P4 Switch and one local compute node

The second node architecture we propose stands for the case where VNF processing functions are demanding and the network should offer an environment with high performance. Since the hardware switch CPU level is limited, we propose having a dedicated high performance compute node collocated with the network switch, as depicted in Figure 2.4.

The characteristics of the compute node are not the object of this document and can be specific to the network operator's needs but one can envisage different options. On the one hand, we have a compute node with very good CPU/GPU/NPU performance, a high bitrate network interface card and a huge memory space. This allows the deployment of any kind of micro-services. On the other hand, one can imagine having a less efficient node, which is less costly, for hosting several minimal micro-services or which do not require specific resources. In the case of more demanding micro-services, the operator can then deploy them in the regional zones (used by several local zones) and configure the routing as necessary, according to the node architecture described in the next section.

In this kind of node architecture, the P4 hardware switch communicates with the compute nodes with the classical network links (e.g., 100 Gb Ethernet or fiber, etc.). Since the node machines are collocated, the latency overhead induced by the network is negligible. This node architecture is more costly than the

previous one since it requires buying one compute node per local zone, but it can host more demanding micro-services and achieve higher performance.

2.3.3 One P4-based hardware switch and several remote compute nodes

The third node architecture consists in deploying the compute nodes remotely, as depicted in Figure 2.5. It addresses a classical use-case where there is no compute node in the local zone and all performing compute nodes are deployed in regional zones (for use by all the local zones). Beyond this, one can imagine that the network operator deploys some compute nodes in some local zones and not in others. In this case, if the first local zone needs a micro-service deployed in the second zone, the routing can be configured such that switch P4 forwards packets to the compute node in the second zone.

In both cases, for the node configuration, the P4 hardware switch will have to forward the packets to a remote compute node. The routing should be configured efficiently to ensure the low-latency required by the global service.

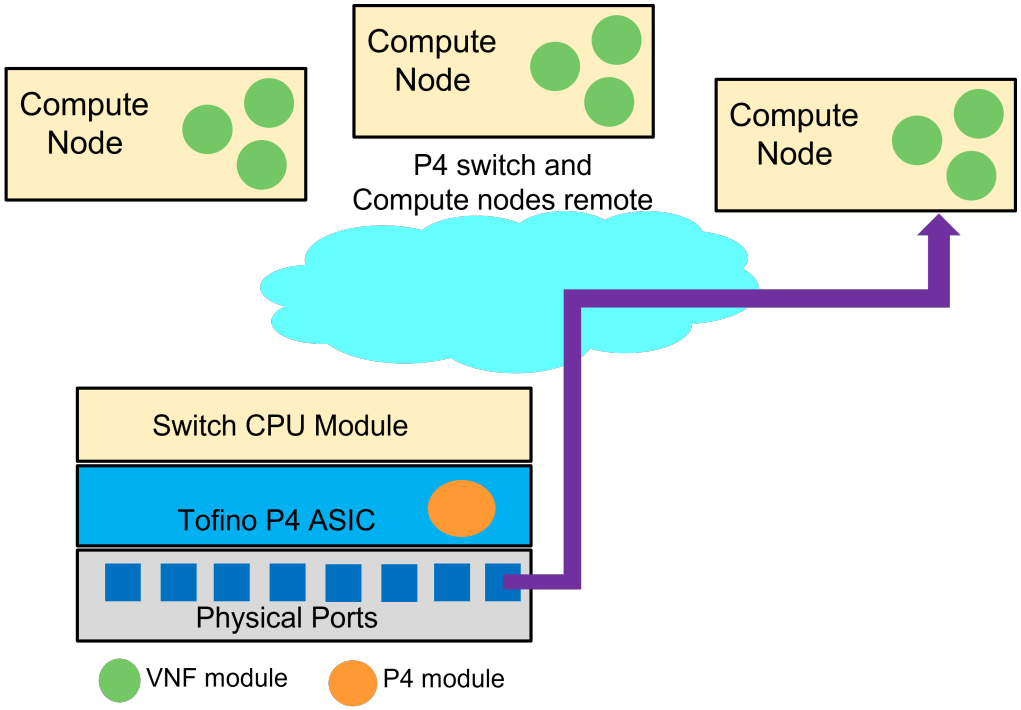

Figure 2.5: MOSAICO Node 3 architecture: P4 Switch and remote compute nodes

2.3.4 Routing between nodes

The MOSAICO project advocates the use of micro-services to form the overall service provided to end users. As such, we designed a two-level MOSAICO node, where packet processing can be done in one or the other level, depending on the function itself, but also on the complexity of the function. Indeed, P4 modules can apply simple processing to packets transiting via the P4 switch, but it cannot handle complex tasks, which then will be done by the application level modules.

This requires a way to compose micro-services and to route packets between them. For this, we envision for instance the use of Segment Routing (SR), based on MPLS labels. This solution is mostly used in networks to set up chains to route packets between network nodes. Here, we advocate its use to route packets according to the service composition, with the orchestration algorithms deciding how to compose the micro-services, and considering different parameters, such as the execution environment, the network topology, the required latency, the nodes' load, etc.

Each micro-service will be assigned with a SR identifier and the global service will be composed by using a stack of SR (MPLS in our case) labels. The first label will be related to the first micro-service to execute, then the second label, etc. until the end of the stack for the final micro-service. When a micro-service is executed, its label is removed from the stack. Figure 2.6 illustrates this concept of Segment Routing.

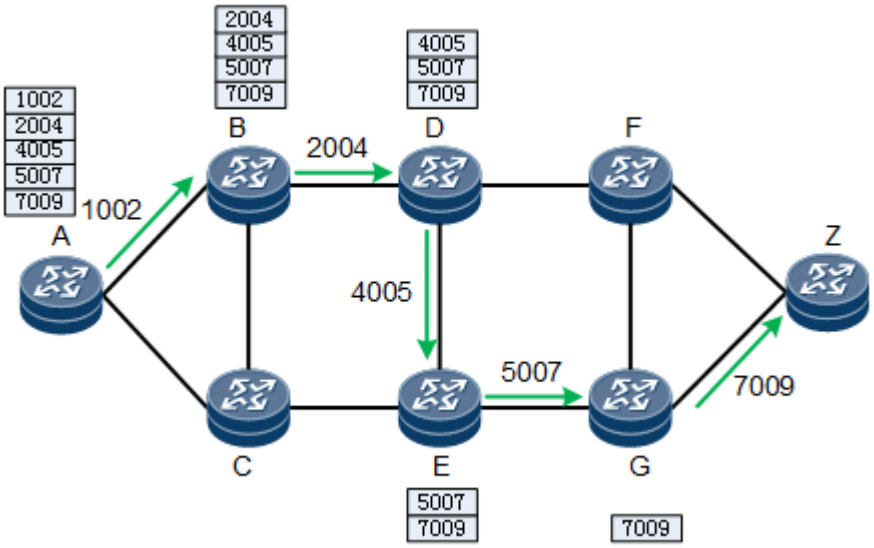

Figure 2.6: Segment Routing Concept

There are consequently two main challenges for our architecture: (1) mixing and routing the execution of micro-services at the two levels (P4 and NFV) and instantiating SR for both levels; (2) manage the removal of SR labels when packets are forwarded from a P4 node to a compute node, which can provide two or more micro-services (e.g., micro-services installed and used within the same compute nodes) since code in the compute node does not have to manage SR labels. One envisioned solution is the use of a proxy-like module in the compute node. This challenge is even more tricky if one micro-service should perform a NAT operation, since we cannot take into account the 5-tuple of the IP connection to keep track of the packet, if several packets from different sessions should be forwarded to the compute node to be processed following the same or a different service chain. A solution using a hash function can be used and this will be addressed in the MOSAICO project.

Chapter 3

Specification of Micro-Services

In recent years, the benefits of micro-services have been demonstrated [3] as an alternative to standard monolithic VNFs which have four important limitations: overlapping functionality, loss of CPU cycles, lack of flexibility in scaling and lack of agility in evolving the code including its testing and integration. The decomposition of VNF into micro-services provides greater flexibility in deployment, the mutualization of certain functionalities and occasionally a reduction in the length of the Service Function Chain (SFC). By their nature, micro-services are lighter and operate autonomously, which makes them more manageable during deployment, opening new possibilities for placement, and possibly allowing operation in parallel, which may minimize the execution time of SFCs. Furthermore, the functional decomposition of VNF allows mutualizing a certain number of identical functionalities within the same SFC, which is not possible in monolithic solutions, and this mutualization would reduce the length of the SFC and therefore the latency.

In the context of the MOSAICO project, we propose different micro-services directly related to the requirements identified, which we detail in the following. Given the parallel scheduling of the project tasks, we have from one side considered different micro-services, in an abstract way (i.e., without implementing them) starting from the literature [18, 14]. This theoretical inventory is conducted to support the work on the placement and routing models, which requires a large number and different types of micro-services to be properly evaluated. We present this early selection of abstract micro-services in the next section. Then, we depict the functional specification of the different micro-services that the project aims at implementing in its following steps. We order them according to their relation with the data-plane (translation of L4S into P4 micro-services, cloud gaming session detector) or their more general monitoring (INT for P4-based L4S) and security (detection requirements) purposes.

3.1 Abstract micro-services for early validation of placement functions

Several micro-service frameworks aimed at network functions have been designed and implemented, such as Microboxes [13] and Openbox [2]. These provide specific performance improvements mainly based on lightweight virtualisation with containers and zero-copy packet processing using DPDK [6]. Requiring an adapted model for their placement and chaining, and with the lack of a specific one, we have proposed in our work [15] a Mixed-Integer Linear Programming (MILP) model that considers the particular features of micro-services and allows an optimised deployment offering net gains in terms of latency compared to monolithic approaches.

To evaluate our model, we consider different SFCs composed of micro-services which reflect those studied in the literature [18, 14]. They are composed of NFs (e.g., Firewall, NAT, Traffic monitor and IPS) whose decomposition into micro-services is known such as those illustrated in Table 3.1. The transition from a monolithic SFC to a micro-service SFC implies a significant increase in the number of entities on the average multiplied per five, which makes its deployment more complicated. For example, if one considers a generic LL service chain composed of a NAT and a Firewall, its decomposition into micro-services, as found in the literature leads to nine micro-services. However, the micro-services induce a lighter footprint in terms of memory space usage as well as computational resources, as compared to monolithic VNFs. Despite this, various tests [15] have shown that a decomposition into micro-services increases the latency of SFCs. Table 3.2 summarises the characteristics of the micro-services used in our

evaluation. In order to benefit from the advantages of the micro-services approach, it is necessary to mutualize and parallelize the micro-services when possible. Figure 3.1 and Figure 3.2 show generic SFCs composed of a NAT and firewall, and its raw decomposition into micro-services. The NAT is functionally decomposed into four micro-services, and the firewall into five. This decomposition makes it possible to observe different overlapping functionalities within the same SFC (e.g., classifier), so that different functionalities can now operate in parallel. Figure 3.3 shows an optimisation of the SFC which allows it to be deployed more easily than in the case of a monolithic SFC. The benefits are that it is shorter in length, requires less space and allows the parallelization of certain functionalities.

VNF	**Micro-service decomposition**
Firewall	Read-HClassfier-Alert-Drop-Output
IPS	Read-HClassifier-PayloadClassifier-Alert-Output
NAT	Read-HClassfier-Modifier-Output
Trafic Monitor	Read-CheckIP-Http Classifier-CountURL-Output

Table 3.1: VNF decomposition into micro-services we consider for the abstract evaluation of the placement and routing algorithm

Parameter	Range or value
VNF	Firewall, IPS, Nat, Traffic monitor
Micro-services	Read (Rd), Header Classifier (HC), Modifier (Md), Alert (Al), Drop (Dp), Check IP Header (CIH), HTTP Classifier (HTC), Count URL (CU), Payload Classifier (PC), Output (Out)
SFC Latency	5-10ms according to the SFC
VNF Proc. latency	4ms
Micro-services Proc. latency	1ms
SFC length	5-14 micro-services

Table 3.2: Evaluation parameters we consider for the abstract evaluation of the placement and routing algorithm

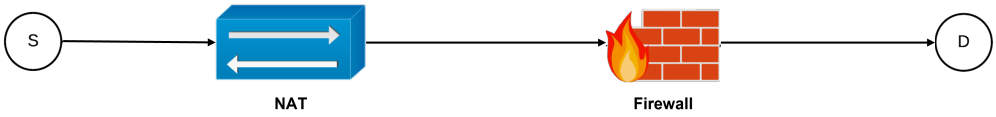

Figure 3.1: Monolithic SFC

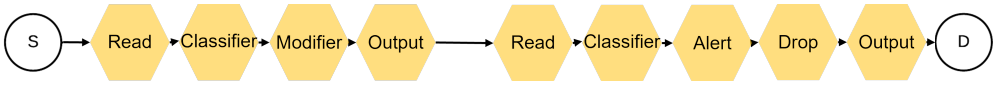

Figure 3.2: Micro-services SFC without optimization

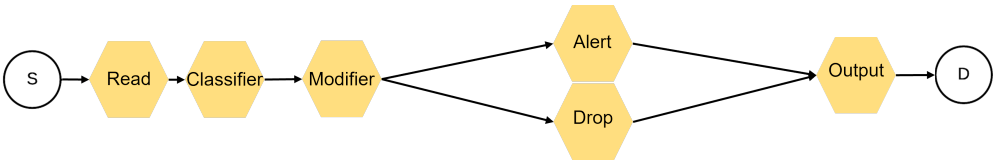

Figure 3.3: Micro-services SFC with optimizations

Finally, regarding parallelization and mutualization, we consider the possibilities identified in the literature, such as [17] and [27], to determine whether two or more micro-services can be mutualized

	Rd	HC	Md	Al	Dp	CIH	HTC	CU	PC	Out
Rd	p,m									m
HC	m	p,m	m	p,m	p,m	p,m	m	p,m	m	m
Md	m		p	p,m	p,m	p,m	m	p,m	m	m
Al	m	m		p,m	p,m	p,m	m	p,m	m	m
Dp	m			p,m	p,m	p,m	m	p,m		m
CIH				p,m	p,m	p,m		p,m		
HTC	m	m		m		m	p,m	m		m
CU	m		m	p,m	p,m	p,m	m	p,m	m	m
PC	m			m	m	m	m	m	p	m
Out	m			m						p,m

Table 3.3: Parallelism and mutualization tables as defined in [17] and [27], with p and m standing for parallelizable and mutualizable micro-services, respectively.

and/or parallelized. In order to know if such optimizations are possible, a static analysis of the different processes carried out between the micro-services in a SFC is necessary. The configuration we consider is depicted in Table 3.3.

3.2 Low-latency forwarding enabler micro-services

In this section, we present two main micro-service use-cases that will be implemented for achieving the low-latency constraints of the cloud gaming use-case. The first one is related to the IETF-defined L4S architecture, where we propose splitting the implementation into micro-services so as to be able to deploy the components at different levels and easily change them without strong dependencies. The second one is a micro-service use-case to detect and classify cloud gaming traffic. This micro-service use-case is one example of a component which can replace an existing L4S IP/ECN classifier. The two micro-service use-cases will then have a link with the data plane, the first one used to forward packets more or less quickly (depending on the LL nature of the sessions or not), the second one analysing the first packets of a sessions before configuring the L4S classifier to forward it via the LL or the classic queue of the L4S system.

3.2.1 Splitting L4S in micro-services

The original implementation of L4S has been done in [4] and is available as open source in [11]. It is written in the C language following a monolithic approach.

L4S can be redesigned according to the micro-services approach we advocate in the project. We propose splitting the L4S architecture into four micro-services: the *classifier*, the *queueing system*, the *AQM computations* and the *packet decision*. These four functions are independent from each other, can be implemented in several ways, and can thus be updated/changed independently from the others.

As an example, the basic L4S solution works with IP/TCP and ECN flags, but we might imagine other transport protocols with different headers (e.g., QUIC or RTP can also support ECN). For this, only the micro-service *classifier* needs to be changed, whereas the three other micro-services can be kept as they are. Similarly, we might implement an improved AQM solution and we just need to update this micro-service. For instance, in the project, we investigate an AQM more suited to time varying network (such as cellular networks) and our own AQM could then replace the classic L4S one in this micro-services-based architecture.

The following describes how the L4S architecture can be split into micro-services (depicted in Figure 3.4) and explains the role of each micro-service:

- The packet *classifier*, implemented in the ingress part assigns the packet either to the LL or Best Effort (BE) queue. In the current version, this module is simple and just parses the ECN bit of the Diffserv field of the IP header to detect if the packet belongs to an "Accurate ECN" flow or not. When ECN-capable applications agree with network operators, such a classification can provide sufficient security. However, it is purely declarative and not protected. Any application/endpoint can cheat and set this bit to "1" in order to be processed by the network as LL traffic even if it is not an LL application. To avoid this, we develop an intelligent classifier which will be

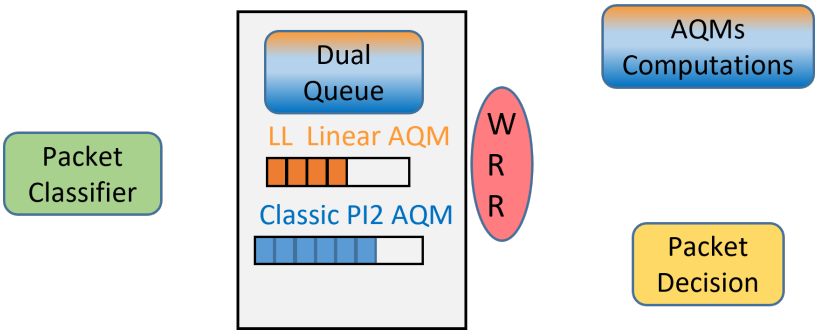

Figure 3.4: L4S splitting into several micro-services

able to analyse and detect the LL behaviour of the sessions transiting via the L4S node, using machine learning methods (e.g., decision tree). Furthermore, since the program is aimed at being deployed in a network equipment processing packets at line-rate, we should implement a fast and efficient solution. Having two possible candidates for this micro-service shows the interest of such a decoupling of the L4S architecture since it allows selecting the most appropriate one, depending on the context.

- The *queuing system* is the part where LL or BE packets will be queued/unqueued to ensure the low-latency requirements of the LL services. In the current L4S implementation, there are two distinct queues, which might be enough. However, in the case of time-varying network capable AQM, one more queue might be needed. This will be refined later and the use of micro-services in this part can also be advantageous.

- The *AQM computations* is the micro-service where the system does the computation of indicators in order to know if the packet should be forwarded, marked or dropped. The current L4S solution uses as input: the time spent by the packet in the queue, the last update time, the last mark/drop probability and the last delay in the queue. This information is required by the PI2 algorithm [4], as well as the PI2 parameters (Update time, α, β). We implement a linear AQM and a PI2 AQM as suggested by the IETF, but when our own time varying network AQM will be designed, we can develop it as a micro-service to replace the current one.

- Finally, the *packet decision* takes the final decision to forward, mark or drop the packets. In our current implementation, following the classic L4S architecture, it is based on the LL and BE computed probability. But we can envision another program taking other information to decide. For instance, for cellular networks, the CQI (Channel Quality Indicator) can be an input parameter to the decision process. This will be refined later when designing our AQM.

3.2.2 Cloud gaming session detector

As illustrated in the previous section related to the split of the L4S architecture into micro-services, we identified that, given our Cloud Gaming (CG) usecase, a session detector, replacing the limited existing one which only leverages Diffserv bits of the IP header is relevant. As such, it is one of the main LL service we investigate in the project with the objective of providing an online traffic classifier which can be inserted into the L4S system. Since, CG traffic is sensitive to latency, we therefore want to recognize it to process it as a LL service. The proposed solution would be deployed at the network edge, where problematic queues can be found due to the *Bufferbloat* phenomenon, when the filling of large buffers incurs large network delays [7].

For each pair of IP addresses, we seek to determine whether a given flow stands for CG traffic or not. To that aim, we propose to rely on a Decision Tree (DT) classifier. The classification is based on 12 features computed over a 33ms time window. These features are obtained from packet sizes and Inter Arrival Times (IATs). Due to the asymmetric aspect of CG traffic, we differentiate between the *client-to-server* and the *server-to-client* direction. The proportion of time windows categorized as CG is used to determine the nature of the traffic between an IP address pair (CG or ¬CG). That proportion is calculated during a certain observation period. Please note that each IP address pair is re-evaluated periodically. This period is yet to be defined.

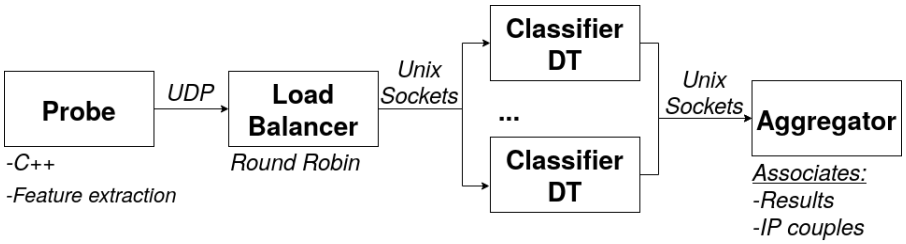

Figure 3.5: Cloud Gaming traffic classifier as a micro-service architecture

For the sake of scalability, we propose to decompose our classification service itself into four different micro-services. We can see on Figure 3.5 an overview of the architecture that we further describe subsequently.

1. *Probe*: listens to a given network interface and computes the features of each IP address pair. Among these features, we find the average of the IATs and the packet sizes, as well as their standard deviation. The other two features are the sum of the packet sizes and their total number. They all relate to a given traffic direction; *client-to-server* or *server-to-client*.

2. *Load Balancer*: listens to the probe's reports. It distributes the load among the compute nodes. This load distribution is done with a *round robin* strategy. The tasks are therefore distributed cyclically between the compute nodes. However, another strategy can easily be applied.

3. *Classifier DT*: a DT is in charge of classifying a time window based on the twelve selected features. This is where CPU cost is the most important. It is therefore essential to classify several time windows in parallel to avoid accumulating delays.

4. *Aggregator*: associates to each IP address pair its number of time windows labeled "CG". Above a certain threshold, a flow can be considered as CG and therefore treated accordingly. The value of this threshold is not yet defined.

Regarding, how those microservices will be implemented (P4 or VNF) and where they will be deployed (local or regional zone) is still to be discussed and evaluated. At least, several options make sense:

- full deployment at the edge;
- probe deployed at the edge and classification deployed in the regional zone;
- full deployment in the regional zone.

The probe is surely the micro-service that would benefit the most from a P4 implementation because it deals with traffic in real time and only computes simple statistics. The classification service could also be implemented in P4 because some previous works have shown that decision trees are simple enough [24] to be implemented in P4 (but not without some adaptations). But the advantage here is less clear and should be carefully evaluated.

3.3 Monitoring and security micro-services

L4S is a novel architecture which exhibits satisfying performance under normal traffic conditions [16], but the question of its capability to deal with abnormal traffic is still an open issue. For instance, the ability of L4S to satisfy LL requirements while maintaining a good balance with Best Effort traffic makes it highly sensitive to non-regular traffic, as illustrated in [1]. This work studied the impact of traffic burstiness on the L4S forwarding performance. Besides, malicious users could easily exploit such weaknesses to pollute the network traffic and degrade the Quality of Experience (QoE) of consumers of LL applications. This is already the case of cloud gaming attacks which, by leveraging Booters [9], are able to target a set of users playing a common match to provoke a poor QoE making the game eventually unplayable.

Generally speaking, the availability of services has always been a concern in networking environments. Legacy volumetric Denial of Service (DoS) attacks [26] have been progressively refined by attackers to obtain a large set of means whose purpose still remains to undermine the service availability to legitimate end users or its proper operation. Low-latency services cannot avoid this and can be subject to attacks

targeting the latency constraints. Particular research and standardization efforts are brought either to intrinsically protect them against such threats or to detect and mitigate them when they occur. For example, Ergenç *et al.* [5] and RFC-9055 [8] propose a comprehensive attack surface identification accompanied by some guidelines for further detection and protection mechanisms focusing on the Time Sensitive Networking (TSN) and Deterministic Networking (DetNet) architectural models [19]. Similarly, the 5G URLLC (Ultra-Reliable Low Latency Communications) group devotes some substantial efforts to ultra-low latency service security [25] since a malicious usage of protocols can lead to quality reduction or a complete denial of the service delivery [10]. However, despite their relevance to their architectural context, these solutions cannot apply to L4S, which is the architectural model we focus on in the MOSAICO project. Consequently, in the following, we provide the functional specification of the micro-services in charge of: (1) monitoring the performance of L4S traffic, by leveraging Inband Network Telemetry (INT), and (2) detecting any abnormal flow which may be a threat to the proper operation of L4S regarding other legitimate flows.

3.3.1 Monitoring L4S using P4-based INT

As a main functional and architectural guideline, the MOSAICO project selected the P4 Working Group specification [22] and it adapts it to its own requirements and especially the Montimage Monitoring Tools (MMT). In this P4 INT reference architecture, one can basically distinguish INT-capable devices, which can be eventually configured by an SDN controller, and a collector, which receives and extracts information from INT reports that are sent by the devices.

Figure 3.6 details the components inside the switches and the collector of our framework to monitor L4S using P4-based INT. We configure the switches via their control planes by using the Thrift protocol. The INT monitoring configuration allows to enable or disable INT capability, to set the INT roles, to select the metrics to be monitored, and to select the IP packets to carry INT metadata. The L4S switch on the left side plays both the source and transit roles to extract and embed INT metadata in INT packets which are then forwarded to the next switch which plays the sink role to send INT reports to the collector.

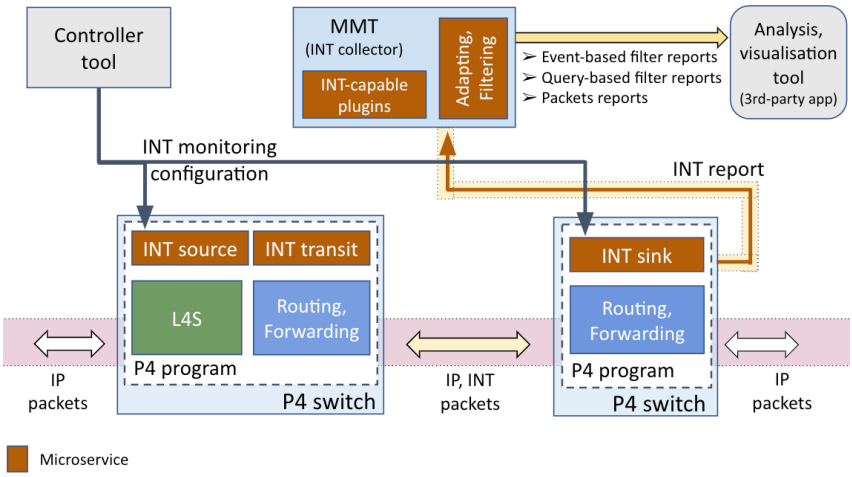

Figure 3.6: Micro-services in L4S monitoring framework using P4-based INT

The micro-services in our monitoring framework are divided into two groups: P4-based INT micro-services and INT collector micro-services. The first group consists of the micro-services that are programmed using P4 language and executed inside the switches together with the L4S micro-services to be able to monitor the L4S's behavior. The second group consists of the micro-services to receive and process the monitoring results generated by the micro-services in the first group.

We further detail the micro-services below:

1. P4-based INT micro-services

   - INT source: it receives from the control plane the list of metrics to be monitored and the masks to select IP packets which will carry INT metadata. It then selects IP packets and embeds a telemetry instruction bit map into these packets to indicate the network information to be measured

- INT transit: while matching and forwarding an INT packet, the transit micro-service interprets the instructions to collect the required information. Then it embeds the information into the packet with respect to the INT metadata protocol as defined in [22] before the packet is forwarded to the next node by the Routing/Forwarding service in the switch.
- INT sink: it extracts the information from the INT metadata and removes them from the packet. Then, it sends the extracted information to the INT collector with respect to the INT report protocol defined in [22].

2. INT collector micro-services

- INT report parser: this micro-service is implemented inside MMT which acts as an INT collector to be able to parse the INT reports sent by the INT sink micro-service.
- INT report filter: it performs a lightweight verification on the INT report parsed results before notifying the third party application, which further provides analysis and visualisation tools.

3.3.2 L4S security detector

Given the security threats briefly exposed above as well as the insights we collected from an exhaustive measurement campaign [12], we identified the need for some reactive security mechanisms for L4S and the way we plan to design, deploy and validate our solution is as follows:

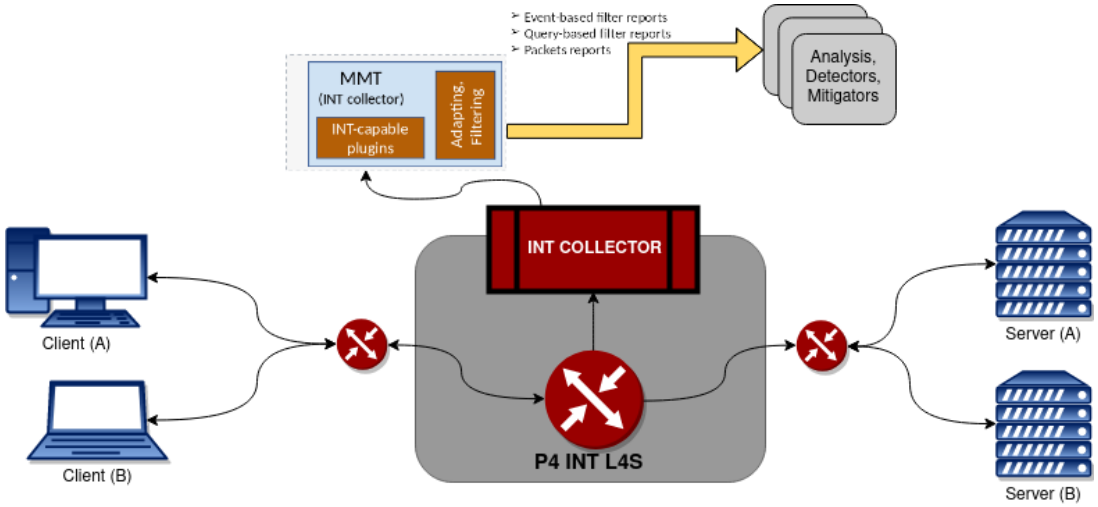

Figure 3.7: L4S testbed instrumented with In-band Network Telemetry

Selecting relevant metrics: The security detector must consider a minimal set of relevant metrics to feature the abnormal phenomena it is willing to detect. A detailed analysis of known threats [12] is giving us some insight about which metrics should be monitored. With the help of a Principal Component Analysis applied to the data collected with In-band Network Telemetry, we aim at selecting key metrics to monitor for the detection of threats targeting low latency requirements. The detector should be robust to various versions of a given threat and, if possible, it should be able to detect sub-RTT variations of both behavior and emitting pattern.

Select a relevant attack among the set of identified undesirable flows: We have previously featured the impact of three types of undesirable flows in L4S: bursty application flows, micro-bursts due to unpaced emitting window management and unresponsive flows. Given the ease for an attack to manipulate ECN based congestion control, we selected the unresponsive ECN attack as our main use case.

Leverage a proper monitoring framework: Standard instrumentation and monitoring are not adapted to the type of phenomena we aim at detecting since their time granularity is not sufficient to provide quick detection and reactions. To that aim, we consider the INT framework exposed above to collect data. The envisioned setup is shown in Figure 3.7. This new way of getting traffic information from our tests will let us access a more fine-grained understanding of what happens within

the AQM during workload pressure. The number of competing flows and the power of the attack will be tested. The power of the attack can be adjusted by letting the malicious emitter behaving like a legitimate flow for a certain period of time. The malicious emitter can possibly manage to misbehave only in short periods of time.

Consider the latency vs. accuracy tradeoff: Several methods will be covered for detecting those undesirable flows. In our approach, two criteria must be respected in order to validate the detector: accuracy and efficiency. Accuracy should be good enough to prevent the detector from impacting legitimate flow, i.e. the false positive rate should be low to not misclassify a legitimate flow and the false negative rate should be close to zero in order to properly handle every undesirable flows. Timeliness is also a important criterion to consider because each millisecond spent on detection is a millisecond that will either be injected in legitimate low-latency flows or be responsible for the delay of reaction: the time spent on detection may not be directly injected in monitored flows, as the detection process may be offline or in parallel by duplicating packets, but in any case, this time means detection delay. A compromise between those criteria should be found in order to define an acceptable level of performance of detection.

Select and compare different ML approaches: Different unsupervised machine learning techniques will be considered and compared to identify those or the one which satisfies with the requirements exposed above. A first offline processing will be achieved to measure the classification performance of each of them. Finally, a selected one will be implemented into the MMT INT framework.

Chapter 4

Some Industrial Trends in Network Function Disagregation

The MOSAICO project considers network function disaggregation as one of the key promising innovations, where micro-services are a key driver for cost reduction and agility in the delivery of LL services while also bringing SFC optimization and ease of placement. Beyond the set of micro-services initially selected by the project as prime candidates for deployment, and our selected use case about the transport of CG traffic on WAN, there is a global trend in applying such disaggregation methodologies in several other contexts. We review them here to guide the project for defining further validation scenarios. The first case considers a 5G scenario with the disaggregation of a base station, while the second case addresses the case of a broadband network gateway.

4.1 Disaggregated base station

The base station is one of the network functions which follows the disaggregation trend. There are many different possibilities to disaggregate the base station. For example, in TR 38.801, the 3GPP identified 8 options (with some sub-options) to split the base-station at various layers of the 3GPP stack, as represented by Figure 4.1.

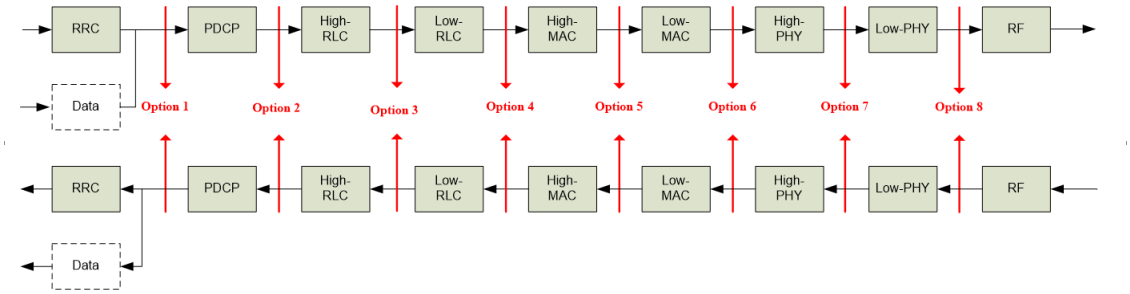

Figure 4.1: Base Station splits identified in 3GPP TR 38.801

Each option brings different bandwidth and latency constraints on the front-haul network connecting the different parts of a split base-station. These constraints were evaluated and summarized in 3GPP TR 38.801 as depicted by Figure 4.2.

Given, the huge number of disaggregation-reaggregation scenarios, the industry is currently attempting to rationalize the number of options into a more limited set. The O-RAN alliance appears to be the main driving force in this consolidation effort. In the O-RAN architecture [20], the disaggregated base-station is re-aggregated into O-Cloud: the cloud infrastructure and the following five logical entities:

- the O-RU for the O-RAN Radio Unit. It contains the Low-PHY (FFT/iFFT, PRACH extraction) according to a 7.2x split and Radio Frequency (RF) processing.
- the O-DU for the O-RAN Distributed Unit. It contains the RLC/MAC/High-PHY layers according to a 7.2x split.

Protocol Split option[1]	Required bandwidth	Max. allowed one way latency [ms]	Delay critical feature[2]	Comment
Option 1	[DL: 4Gb/s] [UL: 3Gb/s]	[10ms]		
Option 2	[DL: 4016Mb/s] [UL:3024 Mb/s]	[1.5~10ms]		[16Mbps for DL and 24Mbps for UL is assumed as signalling]
Option 3	[lower than option 2 for UL/DL]	[1.5~10ms]		
Option 4	[DL:4000Mb/s] [UL:3000Mb/s]	[approximate 100us]		
Option 5	[DL: 4000Mb/s] [UL: 3000 Mb/s]	[hundreds of microseconds]		
Option 6	[DL: 4133Mb/s] [UL:5640 Mb/s]	[250us]		[133Mbps for DL is assumed as scheduling/ control signalling. 2640Mbps for UL is assumed as UL-PHY response to schedule]
Option 7a	[DL:10.1~22.2Gb/s] [UL:16.6~21.6Gb/s]	[250us]		[713.9Mbps for DL and 120Mbps for UL is assumed as MAC information]
Option 7b	[DL:37.8~86.1Gb/s] [UL:53.8~86.1 Gb/s]	[250us]		[121Mbps for DL and 80Mbps for UL is assumed as MAC information]
Option 7c	[DL:10.1~22.2Gb/s] [UL:53.8~86.1Gb/s]	[250us]		
Option 8	[DL:157.3Gb/s] [UL: 157.3Gb/s]	[250us]		

Note: The values are examples provided by LTE reference, as provided in [11] and [14] (modification of required bandwidth in [11]), and are to be replaced by NR values when available. The assumptions for required bandwidth are in Table A-2.

Figure 4.2: Bandwidth and Latency requirements according to Base Station splits in 3GPP TR 38.801

- the O-CU-CP for the Control Plane of the O-RAN Centralized Unit. It contains the Radio Resource Control (RRC) and the control plane part of the PDCP protocol
- the O-CU-UP for the User Plane of the O-RAN Centralized Unit. It contains the user plane of the PDCP protocol and the SDAP protocol
- the Near-RT RIC for the Near Realtime Radio Intelligent Controller which enables near real-time control and optimization of the O-RAN nodes. Compared to previous base-station architecture, it is an additional entity aimed at introducing more near-realtime intelligence and customization in the Radio Access Network (RAN)

O-RUs are typically deployed at cell sites while the O-DU is aimed at being deployed at edge sites. The O-CU deployment location depends on whether the control plane and user plane of the O-CU are actually split or not. In case of monolithic O-CU, it should rather be deployed next to the O-DU in edge sites, while if it is a split O-CU, the O-CU-UP could be deployed at the edge site while the O-CU-CP could be deployed at the regional site. The targeted deployment locations for Near-RT RICs could also be on regional sites.

In rural areas, where population density is low, cell sites are typically selected to maximize the coverage with the lowest number of sites. There is typically no or very limited overlap of coverage between sites. In urban areas, where population density is high, cell sites are denser than strictly required by coverage needs since the site installations are governed by the capacity needed to transport the expected traffic demand (with a given QoS objective). It would be more commonly feasible to cover a part of the zone normally served by a failing site from another one.

Regarding placement questions (i.e., how to select the physical host on which to instantiate a network function): the hosting capacity of each site is addressed at network planning time according to the previously mentioned constraints with no free capacity to host unanticipated network functions instances (i.e., hosting capacity in capillary network sites is not managed in the same way as hyperscaler datacentre hosting capacities). There is no placement consideration. However, in case of failures on site A, re-configuring the O-RU coverage parameters on site B may allow, on some sites (e.g., some urban sites), to cover more of site A (which is also not a placement problem).

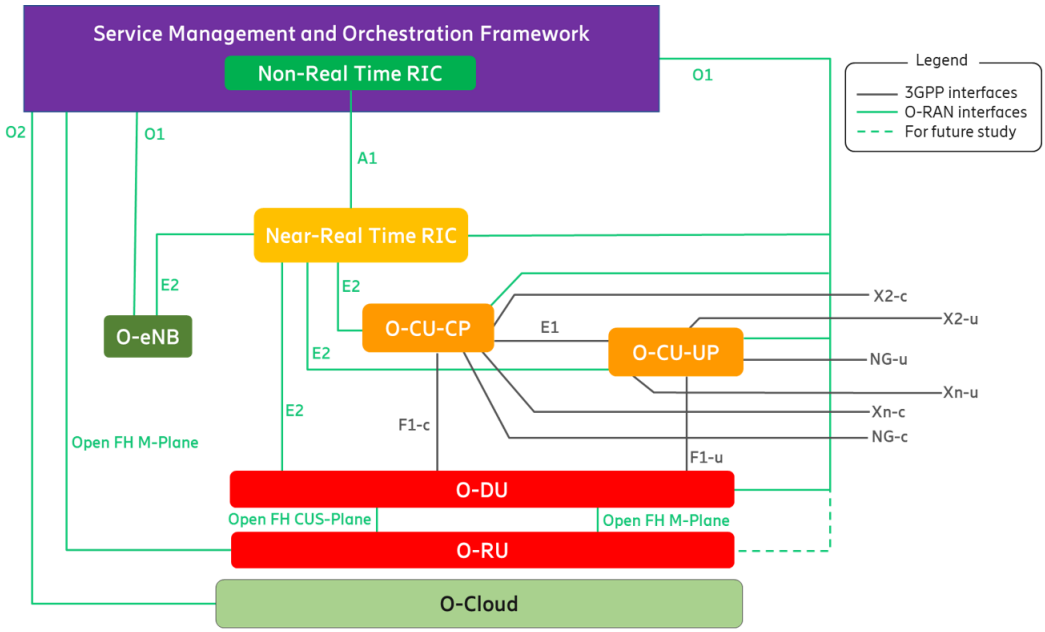

Figure 4.3: O-RAN architecture overview [20]

Given the nature of transmission network topologies, the O-DU, O-CU-UP, O-CU-CP and Near-RT RIC could have one easily identifiable optimal deployment site in terms of latency, determined at network design time. It is possible that multiple edge or regional sites could fit the latency, throughput and processing constraints of these logical entities. Similarly to capillary network, at these network aggregation levels, hosting capacity is still determined at network planning time with usually no free hosting capacity available to instantiate unplanned network function instances.

4.2 Disaggregated broadband network gateway

The Broadband Network Gateway (BNG) is the wireline broadband network function that aggregates the network traffic coming from the digital subscriber line access multiplexers (DSLAM) of a DSL access network and/or from the optical line terminations (OLT) of a passive optical networks (PON) such as a fiber-to-the-home (FTTH) network through an Ethernet backhaul network and historically an ATM backhaul network from DSL customers. Beside (physical) traffic aggregation, it's main functions consist in routing, assisting customer authentication, IPTV multicast, QoS management and accounting. Since, customer identification and customer traffic is simultaneously available on the BNG, it is also often mandated to provide lawful intercept functions. In its so-called Multi-Service variant, the MS-BNG supports the needs of business and wholesale markets (such as MPLS VPN for enterprise customers and wholesale interconnect, such as Layer 2 Tunneling, Ethernet VLANs or IP-VPNs) in addition to the residential market supported by the plain BNG variant.

The broadband forum (BBF) has specified an architecture for the disaggregated Broadband Network Gateway (BNG) in [23] which is depicted on Figure 4.4. As can be seen in Figure 4.4, the BBF only considers the simplest form of network function disaggregation which is the user-plane control-plane split. The BBF identifies the elementary sub-functions of a BNF but it doesn't go beyond this stage, such as the disaggregation into smaller software components. Only the disaggregation implied by the user-plane - control-plane separation is considered, and no software architecture or interfaces are specified for introducing BBF sub-functions.

In the Telecom Infra Project (TIP), an industrial consortium initiated by Facebook with the goal of accelerating the pace of innovation in Telecom by bringing together operators, infrastructure providers, system integrators and other technology companies, the BNG is decomposed into hardware and software components [21]. The software components of the BNG are grouped into four software packages namely: the BNG software package dealing with subscriber management capabilities; the Router software package dealing with its routing capabilities; the Provider Edge (PE) software package to allow the BNG to act as a PE for enterprise services and the BNG CUPS software package for control plane - user plane

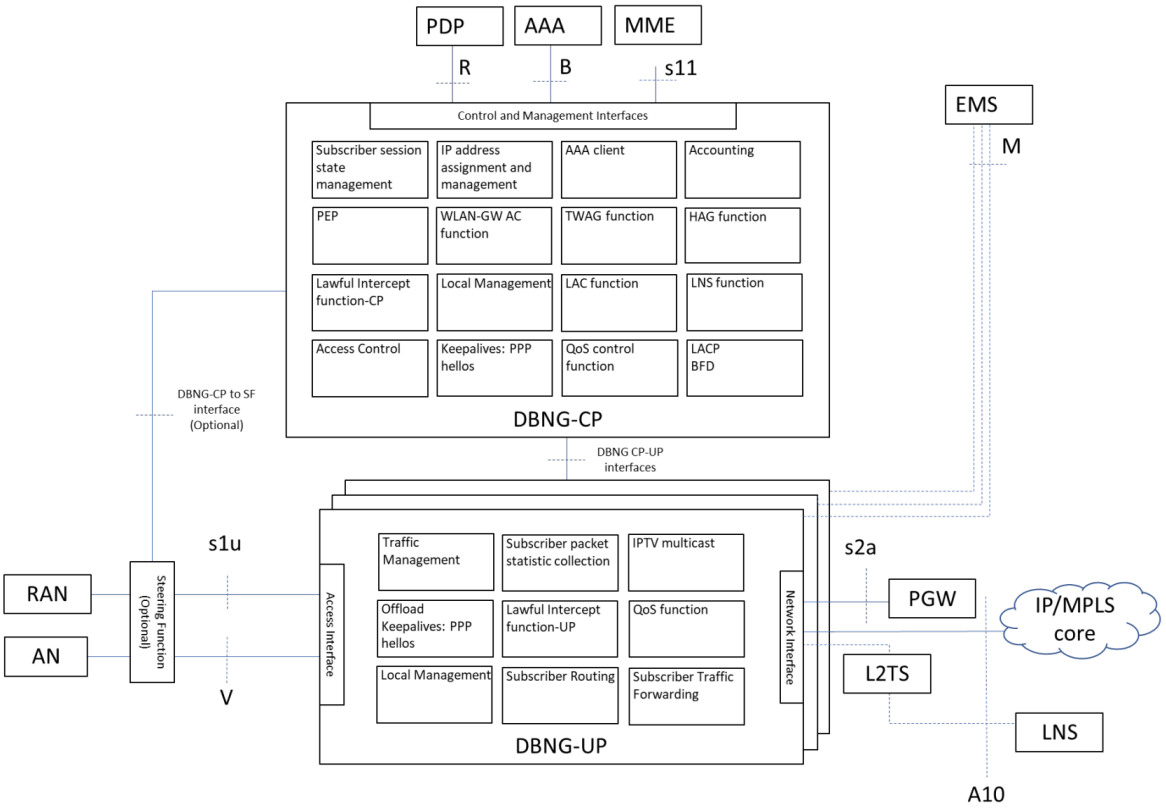

Figure 4.4: Distributed BNG functions separated in Control Plane and User Plane [23]

separation. Although, it's not visible from the software packages as the impact is on the hardware, TIP also considers fixed-mobile convergence with the option to have the BNG acting as an aggregation node of the mobile backhaul which therefore may need to transmit synchronization signals (such as PTP) required for Time-Division-Duplex (TDD) transmission. The option of using the user-plane of the BNG also as the user-plane of an SPGW or UPF (i.e. CUPS for AGF) is left for future work. As shown in 4.5, the software packages grouping more fine grained software components are rather coarse grained, not in the spirit of micro-services. Moreover, the specification document [21] defines the requirements to be fulfilled by each software package but it does not define interfaces between the software packages except interfaces between the user plane and control plane for the BNG CUPS option.

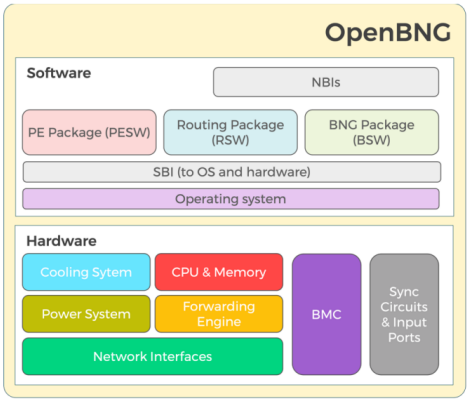

Figure 4.5: TIP's Open BNG platform architecture [21]

Chapter 5

Conclusion

The present delivrable has exposed all the preliminary design choices for the network and node architecture that the MOSAICO project will further explore in the other tasks, and especially Task 4 which focuses on the testbed related activities. In order to benefit from performance, meeting Low-Latency constraints, and also cost reduction and agility, the project followed the network function disaggregation approach which enables the split of functions into micro-services and their deployment at the best topological locations for their mutualization and proximity to the end-user. This guided us toward the specification of a three-layer network architecture composed of local, regional and national levels, corresponding to a realistic telco topology. The project will consider this architecture as a deployment substrate. Considering this topology, we have especially tackled the question of micro-service heterogeneous nature which requires both P4 and lightweight virtualization to be collocated. We have also addressed the case of the micro-service communications within a service chain and selected Segment Routing as a core mechanism.

Given the cloud gaming use-case we previously selected (see D1.1) as a LL service, we have then specified the different micro-services we plan to implement and evaluate in the the project. These are those related to a LL forwarding, following the L4S overall main concepts and architecture, for which we propose a P4 translation, a cloud gaming session classifier which allows to overcome the basic and limit traffic identification performed in the ECN header fields of an IP packet. Then, a dedicated monitoring architecture leveraging Inband Network Telemetry for P4 has also been specified while some early guidelines for the detection of abnormal traffic whose purpose is to undermine the proper delivery of legitimate traffic by L4S have been shaped.

Grounded by this work, the project already started to implement these micro-services and will report them in delivrable D2.2. Some early placement and routing algorithmic solutions are under design with exact method resolution. They will be pushed forward in Delivrable D3.1. Finally, first evaluations obtained from early testbed components are also under way, taking part of the testbed efforts supported by Task 4.

Bibliography

[1] Dejene Boru Oljira et al. "Validating the Sharing Behavior and Latency Characteristics of the L4S Architecture". In: *SIGCOMM Comput. Commun. Rev.* 50.2 (May 2020), pp. 37–44. ISSN: 0146-4833. DOI: 10.1145/3402413.3402419. URL: https://doi-org.proxy.utt.fr/10.1145/3402413.3402419.

[2] A. Bremler-Barr, Y. Harchol, and D. Hay. "OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions". In: *SIGCOMM*. ACM, 2016.

[3] S. R. Chowdhury et al. "Re-Architecting NFV Ecosystem with Microservices: State of the Art and Research Challenges". In: *Network* 33.3 (2019), pp. 168–176.

[4] Koen De Schepper et al. "PI^2: A Linearized AQM for Both Classic and Scalable TCP". In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. Irvine, California, USA: Association for Computing Machinery, 2016, pp. 105–119. ISBN: 9781450342926. DOI: 10.1145/2999572.2999578. URL: https://doi.org/10.1145/2999572.2999578.

[5] Doğanalp Ergenç et al. "On the Security of IEEE 802.1 Time-Sensitive Networking". In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2021, pp. 1–6. DOI: 10.1109/ICCWorkshops50388.2021.9473542.

[6] Linux Sofware Foundation. *Data Plane Development Kit*. URL: https://www.dpdk.org/ (visited on 05/07/2020).

[7] Jim Gettys. "Bufferbloat: Dark Buffers in the Internet". In: *IEEE Internet Computing* 15.3 (2011), pp. 96–96. DOI: 10.1109/MIC.2011.56.

[8] Ethan Grossman, Tal Mizrahi, and Andrew J. Hacker. *Deterministic Networking (DetNet) Security Considerations*. RFC 9055. June 2021. DOI: 10.17487/RFC9055. URL: https://www.rfc-editor.org/info/rfc9055.

[9] Alice Hutchings and Richard Clayton. "Exploring the Provision of Online Booter Services". In: *Deviant Behavior* 37.10 (2016), pp. 1163–1178. DOI: 10.1080/01639625.2016.1169829.

[10] M Awais Javed and Sohaib khan Niazi. "5G Security Artifacts (DoS / DDoS and Authentication)". In: *2019 International Conference on Communication Technologies (ComTech)*. 2019, pp. 127–133. DOI: 10.1109/COMTECH.2019.8737800.

[11] *L4S GitHub*. [Online; last accessed 10/2021]. 2021. URL: https://github.com/L4STeam.

[12] Marius Letourneau et al. "Assessing the Threats Targeting Low Latency Traffic: the Case of L4S". In: *2021 17th International Conference on Network and Service Management (CNSM)*. 2021, pp. 544–550. DOI: 10.23919/CNSM52442.2021.9615534.

[13] G. Liu et al. "Microboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions". In: *SIGCOMM*. 2018, pp. 504–517.

[14] M. C. Luizelli et al. "A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining". In: *Computer Communications* 102.C (2017), pp. 67–77.

[15] Hichem Magnouche, Guillaume Doyen, and Caroline Prodhon. "Leveraging Micro-Services for Ultra-Low Latency: An optimization Model for Service Function Chains Placement". In: *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. 2022, pp. 198–206. DOI: 10.1109/NetSoft54395.2022.9844040.

[16] Bertrand Mathieu and Stéphane Tuffin. "Evaluating the L4S Architecture in Cellular Networks with a Programmable Switch". In: *2021 IEEE Symposium on Computers and Communications (ISCC)*. 2021, pp. 1–6. DOI: 10.1109/ISCC53001.2021.9631539.

[17] Z. Meng et al. "MicroNF: An Efficient Framework for Enabling Modularized Service Chains in NFV". In: *JSAC* 37.8 (2019), pp. 1851–1865.

[18] A. Mouaci et al. "Virtual Network Functions Placement and Routing Problem: Path formulation". In: *IFIP Networking*. 2020, pp. 55–63.

[19] Ahmed Nasrallah et al. "Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research". In: *IEEE Communications Surveys & Tutorials* 21.1 (2019), pp. 88–145. DOI: 10.1109/COMST.2018.2869350.

[20] *O-RAN Architecture Description 7.0*. [Online; last accessed 12/2022]. 2022. URL: https://www.o-ran.org/specifications.

[21] *Open BNG Technical Requirements*. [Online; last accessed 01/2023]. 2021. URL: https://cdn.brandfolder.io/D8DI15S7/at/9k5s2n8xrp6cv537n5kn77rg/TIP-OOPT-Open-BNG-Technical-Requirements-v20-FINAL-GREEN-PUBLIC_ACCESS.pdf.

[22] The P4.org Working Group. *In-band Network Telemetry (INT) Dataplane Specification V2.1*. Tech. rep. 2020, pp. 1–42.

[23] *TR-459 Control and User Plane Separation for a disaggregated BNG*. [Online; last accessed 12/2022]. 2020. URL: https://www.broadband-forum.org/marketing/download/TR-459.pdf.

[24] Zhaoqi Xiong and Noa Zilberman. "Do Switches Dream of Machine Learning? Toward In-Network Classification". In: *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. HotNets '19. Princeton, NJ, USA: Association for Computing Machinery, 2019, pp. 25–33. ISBN: 9781450370202. DOI: 10.1145/3365609.3365864. URL: https://doi.org/10.1145/3365609.3365864.

[25] Takahito Yoshizawa, Sheeba Backia Mary Baskaran, and Andreas Kunz. "Overview of 5G URLLC System and Security Aspects in 3GPP". In: *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2019, pp. 1–5. DOI: 10.1109/CSCN.2019.8931376.

[26] Saman Taghavi Zargar, James Joshi, and David Tipper. "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks". In: *IEEE Communications Surveys & Tutorials* 15.4 (2013), pp. 2046–2069. DOI: 10.1109/SURV.2013.031413.00127.

[27] Yang Zhang et al. "ParaBox: Exploiting Parallelism for Virtual Network Functions in Service Chaining". In: *SOSR*. ACM, 2017, pp. 143–149.