Universidade Federal do Espírito Santo

Rafael Silva Guimarães

Cross-Layer Network Programmability for Expressive and Agile Orchestration Across Heterogeneous Resources

Vitória-ES

2021

CROSS-LAYER NETWORK PROGRAMMABILITY FOR EXPRESSIVE AND AGILE ORCHESTRATION ACROSS HETEROGENEOUS RESOURCES

Rafael Silva Guimarães

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 28 de maio de 2021:

Prof. Dr. Magnos Martinello
Orientador

Moisés Renato Nunes Ribeiro
Coorientador

Prof. Dr. Vinícius Fernandes Soares Mota
Membro Interno

Prof. Dr. Charalampos Rotsos
Membro Externo

Prof. Dr. Leobino Nascimento Sampaio
Membro Externo

Prof. Dr. Daniel Kilper
Membro Externo

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 28 de maio de 2021.

Dedico este trabalho à minha família e amigos.

Agradecimentos

Agradeço primeiramente a minha família, esposa e filhos, meus pais e meus irmãos, que sempre me apoiaram e incentivaram durante toda a minha caminhada no doutorado. Sem eles boa parte de minhas conquistas não teriam ocorrido. Agradeço muita pela minha família, em especial minha esposa Vivianne, por encarar comigo todos os desafios (e não foram poucos) ao longo dessa minha jornada no doutorado. Descobrimos coisas maravilhosas do mundo, sempre juntos, confiando um no outro. Agradeço pelo meu filho Theo, nascido na Inglaterra durante o doutorado (um capitulo a parte em minha tese). Ao meu filho amado, João Pedro, por toda a sua bravura, dedicação em nos ajudar e, principalmente, pelo companheirismo de pai para filho que nós sempre tivemos. Enfim, tenho uma família completa e feliz, e isso é o mais importante. Agradeço imensamente meu orientador, Magnos Martinello, amigo de hoje e sempre, que sempre acreditou e depositou em mim toda a confiança. Ao meu coorientador, Moisés, por toda a dedicação e carinho ao qual ele transmitiu para mim e para todos os seus alunos. A todos os professores do laboratório, em especial para Rodolfo, Vinícius e Giovanni. A professora Ana Locateli por toda a ajuda, atenção e carinho.

Aos meus amigos do LabNERDS, não poderia deixar de esquecer algumas pessoas que foram especiais e que contribuiram muito com a minha jornal, e que aprendi muito com eles, como o Victor, Ricardo, João Henrique, Cristina, e Diego. Pessoas bilhantes e que tiva a oportunidade de trabalhar/estudar. Em especial, ao meu amigo/tiozão Renato, que por uma ironia do destino, contribuimos uma grande amizade em Bristol/Uk e que se perdura ate hoje aqui no Brasil.

Aos meus amigos do IFES Cachoeiro de Itapemirim, agradeço por tudo e principalmente pela amizade, em especial os amigos Bruno, Everson, Daniel e João Paulo e Ricardo Maróquio. Aos meus amigos de Bristol, Rodrigo Stange, Anderson, Kalyani Rajkumar (my best indian friend), Navid Solhjoo (my best iranian friend), Sarah e Darren.

Sou muito grato pela minha familia e amizades que conquistei, e que ainda mantenho, o que me faz ter a plena certeza de que esse é um dos principais pilares da vida: ficar rodeado de pessoas que amamos que nos faz bem e que de alguma maneira nos faz crescer como ser humano.

Resumo

A orquestração pode ser vista como uma cola de interoperabilidade agnóstica de tecnologia que desacopla, entende, oferece suporte e fornece comunicação ponta a ponta com base em uma visão unificada de nuvem de pacotes ópticos sem fio. Rede definida por software (SDN) e virtualização de função de rede (NFV) trazem, como habilitadores, novos paradigmas de rede nos quais prometem melhorar a flexibilidade e a programabilidade por controle centralizado. No entanto, a arquitetura das redes de próxima geração precisa lidar com recursos heterogêneos que geralmente se situam em domínios separados: tempo, frequência e espaço em tecnologias sem fio; fibras ópticas, comprimentos de onda ópticos e portas em ambientes com fio; colocação e recursos de computação nas infraestruturas de nuvem. Esta reengenharia disruptiva das arquiteturas de rede já trouxe recursos-chave como divisão de rede (ou seja, compartilhamento da mesma infraestrutura por meio de diferentes requisitos de serviço), permitindo que as operadoras forneçam conectividade e serviços personalizados e sob medida para cada fatia. Portanto, este trabalho contribui estendendo os paradigmas SDN e NFV, introduzindo a programabilidade de rede de camada cruzada que permite um controle e gerenciamento refinados para suportar uma orquestração expressiva em recursos heterogêneos.

Além das extensões funcionais nos paradigmas SDN e NFV, o processo de orquestração precisa funcionar de acordo para atender às novas dinâmicas de reconfiguração de aplicativos críticos, com demandas de comunicações ultraconfiáveis e de baixa latência. Por exemplo, para garantir a transferência, um modelo de programação é necessário para permitir o controle conjunto de redes sem fio, com fio e na nuvem, acompanhando a mobilidade do usuário, degradação do canal de comunicação e interrupções. Como resultado, a orquestração deve selecionar rapidamente entre os caminhos possíveis na rede subjacente. Isso nos levou a afirmar que o processo de orquestração deve ser sustentado em uma nova proposta de roteamento para atender a recursos rápidos e expressivos na configuração de conectividade ponta a ponta.

A tese apresenta uma nova proposta de roteamento explorando as propriedades do Residue Number System (RNS) que reduzem a carga de gerenciamento da construção de tabelas de roteamento distribuídas, em contraste com as abordagens tradicionalmente baseadas em tabelas que precisam manter e depender de operações de consulta de tabela. Uma abordagem multicast de roteamento de origem baseada em polinômios (M-PolKA) é criada, desenvolvida, implantada e avaliada para permitir a reconfiguração ágil do caminho. A expressividade do M-PolKA é demonstrada ao permitir novas funcionalidades, como duplicação de dados, transmissão / recepção redundante de várias células para oferecer diversidade que aumenta a confiabilidade da comunicação.

Palavras-chave: Next-Generation Networks, Network Functions Virtualization, Software-Defined Networking, Service Function Chaining, Edge computing, 5G paradigm, Source Routing, WiFi, Residue Number System.

Abstract

Orchestration can be viewed as an inter-working technology-agnostic glue that decouples, understands, supports, and provides end-to-end communication based on a unified optical-wireless-packet-cloud view. Software-defined network (SDN) and network function virtualization(NFV) bring, as enablers, new networking paradigms in which promise to improve flexibility and programmability by centralized control. However, the architecture of next-generation networks has to deal with heterogeneous resources that generally sit across separate domains: time, frequency, and space in wireless technologies; optical fibers, optical wavelengths, and ports in wired environments; placement and computing resources in the cloud infrastructures. This disruptive re-engineering of the network architectures has already brought key features like network slicing (i.e., sharing the same infrastructure through different service requirements), enabling the operators to deliver tailored and customized connectivity and services for each slice. Therefore, this work contributes by extending SDN and NFV paradigms introducing cross-layer network programmability that allows a fine-grained control and management for supporting an expressive orchestration across heterogeneous resources.

Besides the functional extensions on SDN and NFV paradigms, the orchestration process needs to perform accordingly to meet the new critical applications' reconfigurability dynamics, such as demand ultra-reliable and low-latency communications. For instance, to ensure handover, a programmability model is required to enable the joint control of wireless, wired, and cloud catching up with user mobility, communication channel degradation, and outages. As a result, the orchestration must select quickly among possible paths in the underlay network. This led us to claim that the orchestration process must be underpinned in a novel routing proposal to meet fast and expressive capabilities in setting up end-to-end connectivity.

The thesis introduces a novel routing proposal by exploring the Residue Number System's properties (RNS) that reduce the management burden of building up distributed routing tables, in contrast to traditionally table-based approaches that have to maintain and rely on table lookup operations. A source routing multicast approach based on polynomials (M-PolKA) is created, developed, deployed, and evaluated to allow agile path reconfiguration. M-PolKA expressiveness is demonstrated by enabling new functionality such as data duplication, redundant transmission/reception from multiple cells to deliver diversity that increases communication reliability.

Keywords: Next-Generation Networks, Network Functions Virtualization, Software-Defined Networking, Service Function Chaining, Edge computing, 5G paradigm, Source Routing, WiFi, Residue Number System.

List of Figures

List of Tables

List of Algorithms

List of Codes

Contents

Chapter 1

Introduction

The emerging of new telecommunication infrastructures has brought the diversity of optical, packet, wireless, and cloud resources, which essentially requires high-level abstraction and control of physical and virtual resources. It poses a new set of challenges to design, deploying, and operating the end-to-end (E2E) communication in which has been commonly a manual and long process [Saraiva de Sousa et al. 2019].

The architecture of next-generation networks requires the management of a large variety of heterogeneous resources across different network infrastructures. Network softwarization [Open Networking Foundation 2014] as well as cloud computing introduce new means for efficient and flexible utilization of their infrastructures through a software-centric service paradigm. They bring new ways in which network operators can create, deploy, and manage their services. However, there is a need to model the end-to-end service and have the ability to abstract and automate the control of physical and virtual resources delivering the service. The coordinated set of activities behind such process is commonly referred to as *orchestration*.

For the sake of network design, there must be an orchestration process mastering the heterogeneous resources that generally sit across separate domains: time, frequency, and space in wireless technologies; optical fibers, optical wavelengths, and ports in wired environments [Fu et al. 2018]; placement, and computing resources in cloud infrastructures [Bogue 2017]. In this thesis, we refer to orchestration as the coordination of resources and services embracing *a single administrative domain* composed of its heterogeneous technology footprints. In one administrative domain, multiple technology domains co-exist based on the type of technology in scope, for example, Wireless, Optical Networks and Cloud.

Figure 1.1 highlights how the orchestration works as an inter-working technology-agnostic glue, as an analogy to IP in the traditional network protocol stack. This hourglass shape with a central orchestration framework decouples, understands, supports, and provides a better E2E communication. The strategic role is based on

a unified view considering : (*i*) Software Defined Networking (SDN) orchestration by interacting with optical network (SDON), wireless (SDR) and packet (P4) ; (*ii*) Network Functions Virtualization (NFV) and Cloud orchestration capabilities, such as handover, or automatic scaling of Virtualized Network Function (VNF) based on monitoring of resource usage with VNF lifecycle management (*e.g.,* start, stop, migrate).

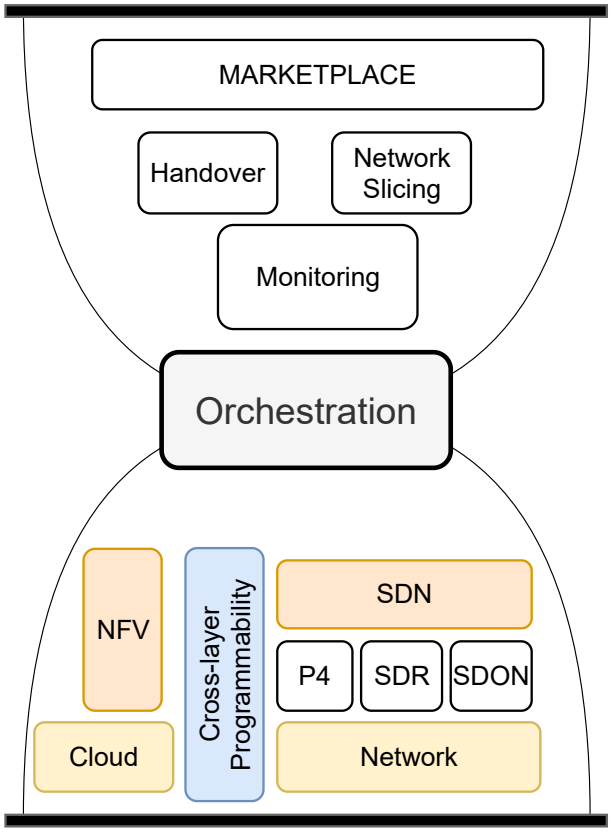


Figure 1.1: Strategic role of the orchestration as the glue between the actual services and the underlying management of resources. Adapted from [Saraiva de Sousa et al. 2019].

Although SDN and NFV have promised high flexibility and new programmability models, SDN was not designed to deal with the configuration of network physical layer. Actually, there is a crucial need to create a broader number of functionalities for the control plane and new capacities to the data plane to drive each resource specificity with its fine granularity control (i.e., time, frequency, wavelength, and space). NFV lacks uniform management of heterogeneous resources, especially when different virtual infrastructure managers have been deployed. An NFV uniformization foresees a high level of virtualization to decouple the software from the underlying hardware to abstract physical resources [Both et al. 2019].

It is worth mentioning that investigations in the state-of-the-art have tackled issues orbiting around the orchestration of optical, wireless, or cloud networks addressing their resources separately [Tzanakaki et al. 2017]. However, this thesis

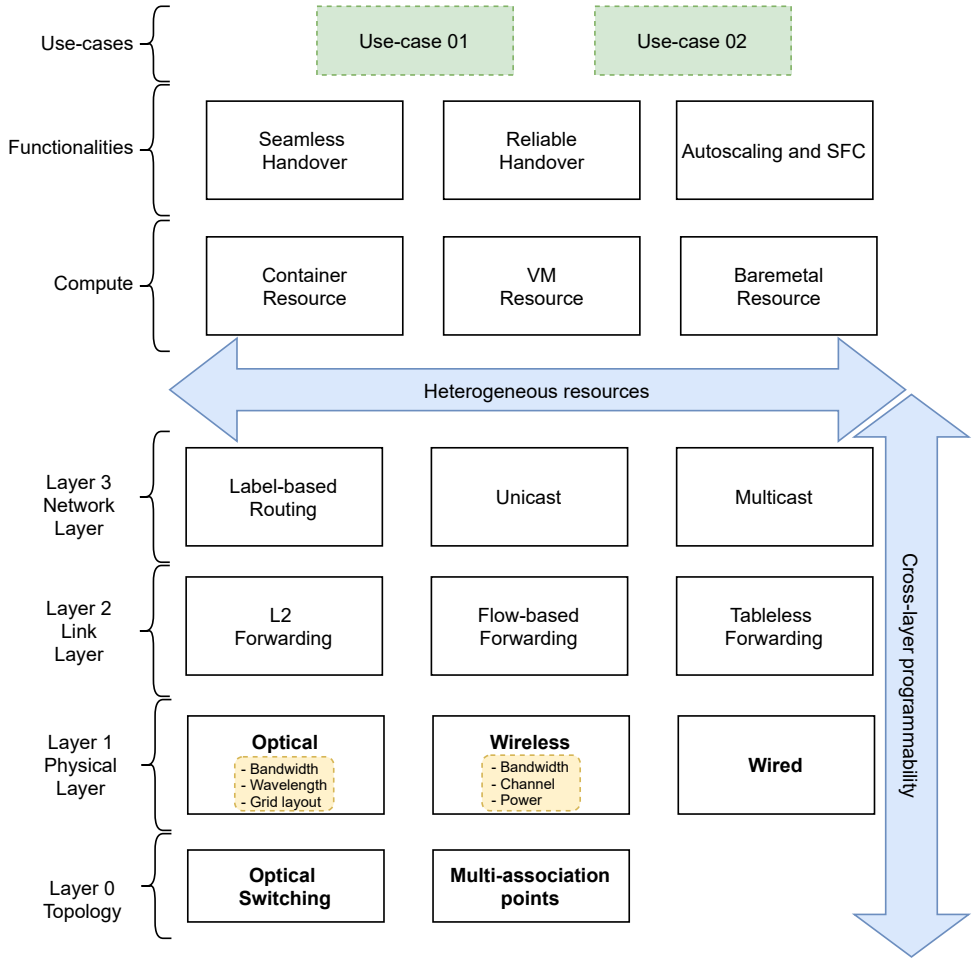


Figure 1.2: Positioning cross-layer programmability and heterogeneous resources.

differs from all other existing works by introducing a unique approach that relies on an integrated orchestration of wireless, optical, packet, and cloud network resources. Hence, this thesis argues that the orchestration process will emerge naturally as network owners take control of their software and engage in open-source efforts due to deep programmability across the stack, both vertically (control and data plane) and horizontally (end to end) [Foster et al. 2020].

Figure 1.2 shows an overview of our approach that extends a range of configurable physical layer (*e.g.*, wireless and optical channels with adaptive bandwidth), link layer (*e.g.*, flow-based forwarding, tableless forwarding, etc;) so that these configurable parameters are exposed to create a cross-layer programmability abstraction (vertical arrow). This abstraction is under the command of an orchestration process that provides functionalities (e.g. autoscaling) so that it may set parameters across heterogeneous resources by using a cross-layer network programmability.

Yet, in terms of orchestration requirements, the orchestration system has to *provide a high expressiveness to set up devices and integrated them into a common collection of functionalities*, which differs from the traditional monolithic behavior.

This is a second requirement to support an *expressive orchestration across heterogeneous resources*. In high mobility scenarios, such as robots in Industry 4.0, there is a stringent requirement for a fast reaction involved in a datapath configuration. Such agility demands quick handovers that require to handle with a large amount of *distributed data plane states*. This time to set the distributed states of the network elements can significantly penalty the convergence time, affecting the reliability during the handover task [Soliman et al. 2012, Heller et al. 2012]. This led us to claim that the orchestration process must be underpinned in a novel routing proposal to meet fast and expressive capabilities in setting up datapath.

Thus, this work also introduces a novel multipath routing, named *M-PolKA (Multipath Polynomial Key-based Architecture)*, mechanism founded on Residue Number System (RNS) encoding, with polynomial arithmetic using Galois field (GF) of order 2 [Shoup 2009], known as GF(2). The proposed scheme M-PolKA builds a tree allowing a rapid configuration at the source. It generalizes a previous work named PolKA [Dominicini et al. 2020], which focused on source routing for single path. Hence, M-PolKA extends PolKA's coding representation, architecture, and deployment performed in an continental testbed [1].

This mechanism can significantly alleviate the pressure on data-plane resources and the overhead on the control-plane during some events (i.e., mobility, failure, and performance degradation). Thus, it allows the exploitation of path diversity to deliver an expressive orchestration capability across heterogeneous resources, enabling fast rerouting, agile path selection, packet duplication, and fast failure recovery.

1.1 Research Question

To tackle the aforementioned problems, this dissertation aims to answer the following question:

- Which are the mechanisms required to enable network programmability to support an expressive and agile orchestration across heterogeneous resources?

1.1.1 Hypotheses

In this work, we envision the following hypotheses to tackle the described problem :

- **Hypothesis H1:** the SDN and NFV paradigms ought to be extended to give a functional cross-layer control and management over heterogeneous resources.

[1] https://wiki.geant.org/display/RARE/Home

  - **Justification H1**: Since the SDN architecture was not conceived to deal with physical layer, this barrier needs to be overcome to achieve the programmability of physical parameters. For NFV, the barrier is about crossing heterogeneous resources that have different virtualized infrastructure managers (VIMs). Therefore, the orchestration process taking advantages of SDN and NFV extensions need to be developed to coordinate what, how and when to set a variety of resources in a single administrative domain. This abstraction should provide functionalities so that it may set parameters for heterogeneous resources horizontally and vertically express them by a cross-layer network programmability.

- **Hypothesis H2:** A source routing mechanism is fundamental to guarantee an expressive and agile orchestration of multipath routing across heterogeneous resources.

  - **Justification of H2**: New critical applications, for example cloud robotics, bring specific requirements such as *zero mobility interruption time*. Despite the SDN centralized control, there is still the need to set devices up individually, along distributed switches that belong to a given path. Usually in a SDN architecture, this is performed by using flow modification messages sent to the switches for providing a route to a mobile object. However, this approach is far from suitable in scenarios where we have high dynamic clients, since the network is not agile enough for datapath configuration. In fact, it demands to handle with a large amount of distributed data plane states. Our premise is that the required agility might be achieved if a tree is built over a stateless core with multipath mechanism configurable at the source edge.

1.2 Objectives

To the best of our knowledge, we have not seen in the state-of-art, an orchestration covering the following problems:

- Firstly, we define the Specific Objective **SO1**. To design and develop an orchestration approach that requires cross-layer network programmability allowing the management and control of heterogeneous resources allocated from wireless, optical, packets, and cloud.

- The performance requirement of the orchestration is the **SO2**, which means that it should support a reliable seamless handover and failover schemes fostering zero mobility interruption time (MIT).

- Lately, the orchestration must be highly adaptable and flexible in setting devices communication which is **SO3**. This objective leads us to introduce a novel multipath routing mechanism founded on RNS encoding. The proposed scheme can express a multi path (a tree) that traverses the network technologies allowing a rapid configuration at the source, even if the resources are heterogeneous.

1.3 Contributions

This thesis makes two fundamental contributions:

1. Extents the SDN/NFV paradigms to provide cross-layer programmability across heterogeneous resources, deploying and evaluating in a real testbed;

2. Introduces a novel source routing multipath mechanism, deploying and evaluating in both emulated and intercontinental testbed.

This thesis was idealized through discussions and studies carried out in the LabNERDS-UFES [2]. Figure 1.3 gives an overview of how the author's publications are mapped to the different areas. While this thesis presents in reverse chronological order, separated by conference and journal. The following contributions bring some clarification about the author's collaboration on each work.

1. List of conference papers:

    - **Publication A**: **Guimaraes, R. S**, Martinez, V. M. G., Mello, R. C., et al. "An SDN-NFV Orchestration for Reliable and Low Latency Mobility in Off-the-Shelf WiFi", 2020, IEEE International Conference on Communications. **Qualis A1**.
    - **Publication B**: Frank Slyne, **Rafael S. Guimaraes**, Yi Zhang, Magnos Martinello, Reza Nejabati, Marco Ruffini, and Luiz A. DaSilva, "Coordinated fibre and wireless spectrum allocation in SDN-controlled wireless-optical-cloud converged architecture", 2019, European Conference on Optical Communication. **Qualis B1**.
    - **Publication C**: Martinez, V. M. G., **Guimaraes, R. S.**, Mello, R. C., Hasse, P., Ribeiro, M. R. N., Martinello, M., and Frascolla, V. (2018). "Ultra Reliable Communication for Robot Mobility enabled by SDN Splitting of WiFi Functions", 2018, IEEE Symposium on Computers and Communications. **Qualis A2**.

[2] http://nerds.ufes.br/en/

2. List of journal papers:

    - **Publication D**: Both, C., **Guimarães, R. S.**, Slyne, F., Wickboldt, J., Martinello, M., Dominicini, C., Dasilva, L. (2019). “FUTEBOL Control Framework: Enabling Experimentation in Convergent Optical, Wireless, and Cloud Infrastructures“, 2019, IEEE Communications Magazine. **Qualis A1**. Impact factor: **11.05**.
    - **Publication E**: Carmo, A. P., Vassallo, R. F., Queiroz, F. M., Picoreti, R., Fernandes, M. R., Gomes, R. L., **Guimarães, R. S.**, Martinello, M., Simeonidou, D. (2019). “Programmable intelligent spaces for Industry 4.0: Indoor visual localization driving attocell networks“, 2019, Transactions on Emerging Telecommunications Technologies. **Qualis B1**. Impact factor: **1.594**.
    - **Publication M**: Dominicini, C. K., **Guimarães, R. S.**, Mafioletti D., Martinello, M., Ribeiro, M. R. N., Villaça, R., Loui, F., Ortiz, J., Slyne, F., Ruffini, M., Kenny, E. “Deploying PolKA Source Routing in P4 Switches“, 2021, **Under revision**, International Conference on Optical Network Design and Modelling (ONDM). **Qualis A4**.
    - **Publication N**: **Guimarães, R. S.**, Dominicini, C. K., Martínez, V. M. G., Xavier, B. M., Mafioletti, D. R., Locateli, A. C., Martinello, M., Ribeiro, M. R. N. “M-PolKA: Multicast Polynomial Key-based Source Routing for Reliable Communications“, 2021, **Resubmition** has been planed to September 2021, IEEE Transactions on Network and Service Management. **Qualis A1**. Impact factor: **3.878**.

The following publications (listed in reverse chronological order) are not directly linked to this thesis. However, they helped to gain important knowledge in the matter and, therefore, should also be considered relevant contributions.

1. List of conference papers:

    - **Publication F**: Rodrigo S Tessinari, Anderson Bravalheri, Emilio Hugues-Salas, Richard Collins, Djeylan Aktas, **Rafael S. Guimarães**, Obada Alia, John Rarity, George T Kanellos, Reza Nejabati, Dimitra Simeonidou, “Field Trial of Dynamic DV-QKD Networking in the SDN-Controlled Fully-Meshed Optical Metro Network of the Bristol City 5GUK Test Network“, 2019, European Conference on Optical Communication, ECOC 2019. **Qualis B1**.
    - **Publication G**: R. Nejabati, R. Wang, A. Bravalheri, A. Muqaddas, N. Uniyal, T. Diallo, R. Tessinari, **R. S. Guimarães**, et al, “First Demonstration

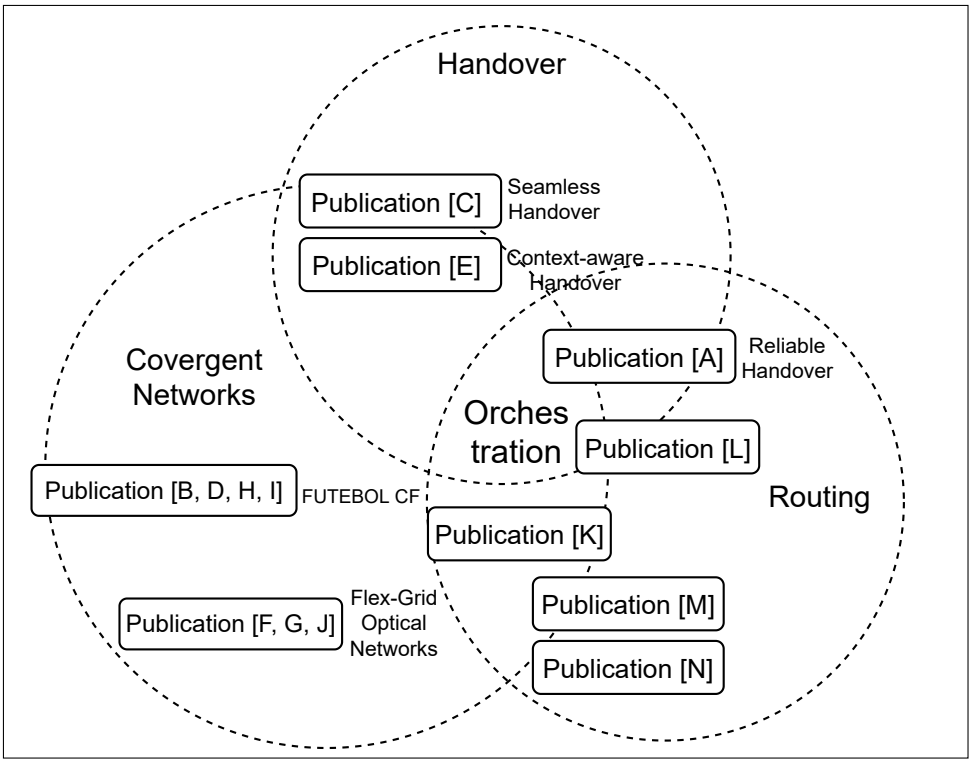


Figure 1.3: Correlation between the main topics and publications of the thesis with the intersection between them.

of Quantum-Secured, Inter-Domain 5G Service Orchestration and On-Demand NFV Chaining Over Flexi-WDM Optical Networks“, 2019, Optical Fiber Communications Conference and Exhibition (OFC), San Diego, Post Deadline, OFC 2019. **Qualis B1**.

- **Publication H**: Paulo Marques, Alexandre P do Carmo, Valerio Frascolla, Carlos Silva, Emanuel DR Sena, Raphael Braga, João Pinheiro, and et al. “Optical and wireless network convergence in 5G systems–an experimental approach“, 2018, IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD).
- **Publication I**: Frascolla, V., Colman-Meixner, C., Nejabati, R., Simeonidou, D., Slyne, F., Zhang, Y., **Guimarães, R. S.**, and Ribeiro, M. R. N.. “When optical networks meet wireless systems: experiments at the boundary“, 2018, In 2018 Photonics in Switching and Computing (PSC).
- **Publication L**: Xavier, B. M., **Guimarães, R. S.**, Comarela, G., Martinello, M.,“Programmable Switches for in-Networking Classification“, 2021, IEEE International Conference on Computer Communications (INFOCOM). **Qualis A1**.

2. List of journal paper:

- **Publication J**: R. Wang, R. Tessinari, E. Hugues-Salas, A. Bravalheri, N. Uniyal, A. Muqaddas, **R. S. Guimarães**, T. Diallo, S. Moazzeni, Q. Wang, G. Kanellos, R. Nejabati, and D. Simeonidou, "End-to-End Quantum Secured Inter-Domain 5G Service Orchestration Over Dynamically Switched Flex-Grid Optical Networks Enabled by a q-ROADM", 2020, Journal of Lightwave Technology. **Qualis B2**. Impact factor: **4.288**.
- **Publication K**: Renato S. Silva, Carlos C. Meixner, **Guimarães, R. S.**, et al. "REPEL: A Strategic Approach for Defending 5G Control Plane from DDoS Signalling Attacks", 2020, IEEE Transactions on Network and Service Management. **Qualis A1**. Impact factor: **3.878**.

1.4 Text Structure

The remainder of this thesis is divided as follows:

Chapter 2 presents in more detail the comparation with related works. It covers crucial concepts related to network softwarization (SDN and NFV), seamless and reliable mobility in next-generation networks (5G).

Chapter 3 gives an overview of orchestration architectures for next-generation networks. This Chapter also presents i) the design and implementation of our orchestration architecture to support the unified control of physical parameters for optical, wireless, and cloud domains to guarantee a better expressiveness; and ii) a solution enabling cross-layer programmability to the orchestrator.

Chapter 4 gives an overview the strict requirements on reliable and low-latency communication of critical services. The Chapter also proposes an orchestration architecture to ensure reliable and low-latency handover mobility with zero MIT across heterogeneous resources. Finally, the Chapter prototypes the solution in a proof-of-concept testbed for functional and performance tests in a closest-to-production environment, and discuss about the obtained results and conclusion remarks.

Chapter 5 gives an overview of the benefits regarding the use of source-routing for fast reconfiguration and how it can be suitable for critical services in the next-generation networks. Chapter 5 also fosters the next steps of our architecture for a new residue number system that can provide a fast reconfiguration through heterogeneous resources aiming reliable and low-latency handover solution in WiFi and Cloud contexts. The Chapter finishes with a case study and discussions about the obtained results and future works.

Chapter 6 concludes this Thesis discussing about the next steps and future works.

Chapter 2

Foundations and State-of-the-Art

This chapter describes some of the fundamental background and key concepts that are relevant to understand this work. We discuss the network softwarization principles and foundations including Software-Defined Networking (SDN), Network Function Virtualization (NFV), Management and Orchestration.

2.1 Software-Defined Networking

SDN is an enabler of a network softwarization architecture based on the following characteristics: (i) the separation between the control plane and data plane; (ii) uniform and vendor-independent interface to the forwarding engine;(iii) logically centralized control plane, and (iv) capability of virtualizing the underlying physical network [McKeown et al. 2008] [Open Networking Foundation 2014].

The control plane acts as a networking operating system, which observes the entire state of the network from a central point and controls the forwarding plane. The idea is that a controller centralizes the communication with all programmable network elements, and offer a unified view over the whole state of the network [Casado et al. 2010]. Being this entire view, the controller can proceed with the forwarding logic, where data plane acts. Hence, the data plane defines the behavior of each forwarding element (switch, router, access point, or base station), hosting features such as routing protocols, access control, network virtualization, and energy management [Lantz et al. 2010].

Indeed, one of the great benefits of SDN is the ability to use this centralized view to perform analysis and make custom decisions about the overall system operation and the configuration of each network element. It is important to note that the controller is logically centralized. However, it can also implement in a distributed way by dividing the control elements between different domains or in a truly distributed controller in which, for example, uses accord algorithms to

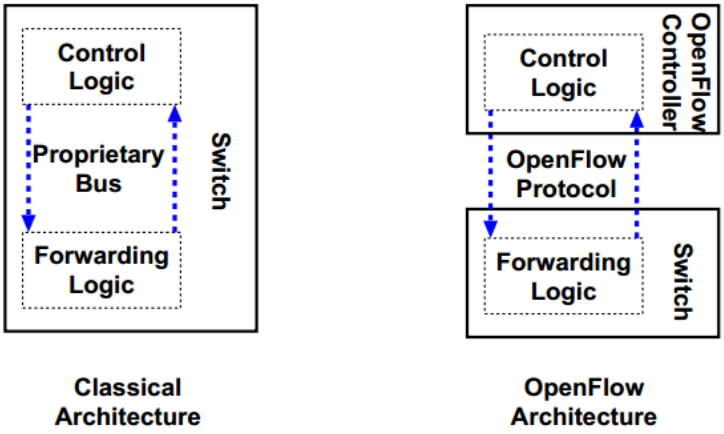


Figure 2.1: Traditional networking architecture versus OpenFlow architecture. Source: [Sherwood et al. 2009].

consolidate a view among its parts [Guedes et al. 2012].

Further advantages of SDN is that network functionalities can open the innovation of networks by the deployment without modifying the hardware, allowing the network to evolve at the same speed as software, rather than to wait for the development of standards and new hardware. As a particularly exciting benefit to DCNs is that a centralized controller makes high-level packet manipulation decisions, so that the data plane can be implemented in low-cost commodity switches [Verdi et al. 2010].

2.1.1 OpenFlow

The OpenFlow standard is the most successful and widely adopted implementation of the SDN paradigm. Its primary purpose is to provide a standard and open programming interface that allows remote control of routing tables of forwarding devices such as switches, routers and access points.

In [Sherwood et al. 2009], they compare a classical network architecture, where control and data forwarding logics are located at the same device (for example, a switch) and communicate through an internal proprietary bus, with the OpenFlow architecture. The control logic is transferred to an external controller that communicates with the device responsible for forwarding logic using the OpenFlow protocol. Figure 2.1, shows in more details this difference.

According to the OpenFlow architecture, an OpenFlow switch has at least three components [McKeown et al. 2008], as shown in Figure 2.2:

- The flow table, which has rules associating a flow by a match and action, hence, telling the switch how to process each flow.
- The Secure Channel that connects the switch to the controller, allowing the

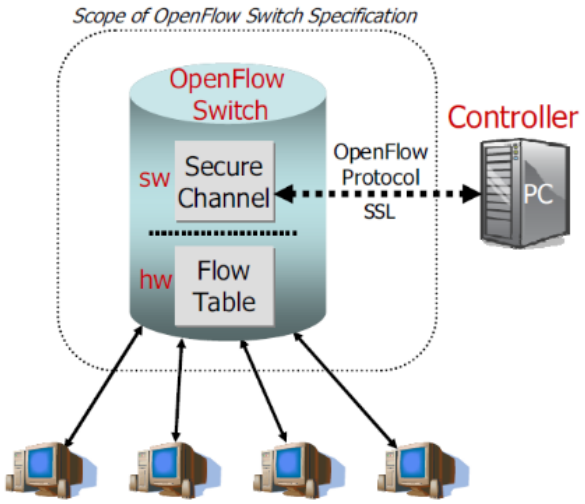


Figure 2.2: OpenFlow architecture. Source: [McKeown et al. 2008].

signaling between those planes sent with confidentiality and integrity.

- The OpenFlow Protocol, which provides standardization in an open protocol for a controller to communicate with the switches. It also allows entries in the flow table to be defined externally.

By abstracting forwarding rules as flow entries, the OpenFlow protocol calls the collection of flow entries in the data as flow tables. Each flow table entry correlates a rule with a flow, and, if a packet fits into a specific rule, the defined actions are taken [Guedes et al. 2012] (e.g., forward the packet to a specific port, rewrite part of its headers, or forward it for inspection in the network controller). If there is no rule defined for a given packet, the data plane sends the message to the controller and takes some decision (i.e., in most of the cases, the controller defines a new flow entry). In this approach, these rules allow remote control of how packets that are part of a specific flow must be forwarded and processed in each network element, according to the interests identified by the centralized view of the controller.

2.1.1.1 OpenFlow Controllers

As earlier mentioned, that evolving networking paradigm, an OpenFlow controller manipulates the flow table entries in the network elements through the OpenFlow protocol and can act statically or dynamically [McKeown et al. 2008]. In the case of a static controller, the controller may be, for example, a simple application running on a PC that statically establishes a set of flow entries to enable the interconnection of some test computers during an experiment. In the case of more sophisticated controllers, the controller can add and remove flow entries dynamically during network operation and even allow different users and applications to interfere with control decisions. There are currently a number of controllers that support the OpenFlow

protocol, such as: Floodlight [1], Ryu [2], ONOS[3], and OpenDaylight[4]. From this perspective, an orchestration coordinates multiple SDN controllers that recognize the needs of higher-level orchestrator(s). Therefore, SDN controllers can be programmed to monitor the network and make automated (real-time) decisions in case of security problems, faulty devices, traffic congestion [Saraiva de Sousa et al. 2019].

2.1.1.2 Open vSwitch (OvS)

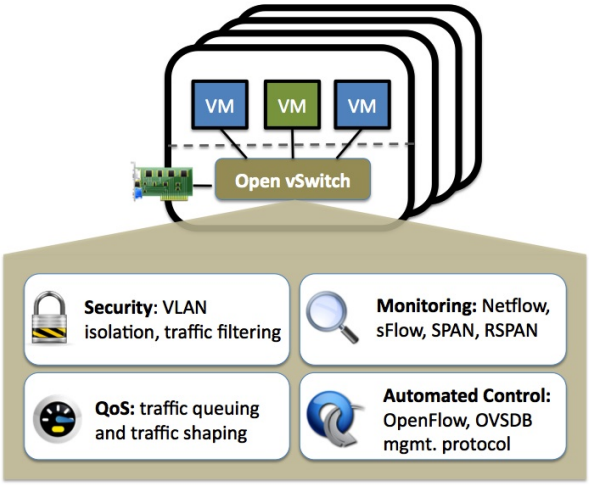


Figure 2.3: Open vSwitch overview. Source: [Openvswitch.org 2015].

Open vSwitch (OvS) is the most popular open-source software switch that supports a wide range of OpenFlow protocol versions and management interface. It delivers a high production quality switch platform and opens the forwarding functions to programmatic extension and control. It has features comparable to physical switches, but with the flexibility and speed of development. Thus, it facilitates the network management in virtual environments, as well as offers additional functionalities not available hardware switch commodities [Verdi et al. 2010]. The OvS offers improved performance for OpenFlow architecture by separating the data plane into Kernel level from the Linux operating system. In contrast, the control plane is accessed from the user space [Guedes et al. 2012] and gives us a powerful tool for implementing network virtualization using the OpenFlow protocol. It was designed to supports multiple Linux-based virtualization technologies, including Xen/XenServer, KVM, and VirtualBox.

[1] http://www.projectfloodlight.org/floodlight/
[2] https://osrg.github.io/ryu/
[3] https://onosproject.org/
[4] https://www.opendaylight.org/

2.1.2 P4: Programming Protocol-Independent Packet Processors

P4 is a high-level language for programming protocol-independent packet processors. P4 expresses how the packets are processed by the data plane of a programmable forwarding element such as software switch, network interface card, router, or network appliance. P4 can also work in conjunction with SDN control protocols like OpenFlow. The newest version is P416 that makes many significant, backward-incompatible changes to the syntax and semantics of the language. Therefore, P4 programs specify how the various programmable blocks of a target architecture are programmed and connected. The Portable Switch Architecture (PSA) is a target architecture that describes standard capabilities of network switch devices that process and forward packets across multiple interface ports. The PSA Model has six programmable P4 blocks and two fixed function blocks, as shown in Figure 2.4. The behavior of the programmable blocks is specified using P4 language. The Packet buffer and Replication Engine (PRE) and the Buffer Queuing Engine (BQE) are target-dependent functional blocks that might be configured for a fixed set of operations [P4 v16 2020].

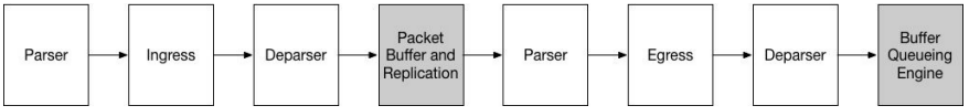


Figure 2.4: Portable Switch Pipeline. Source: [P4 v16 2020].

2.2 Network Function Virtualization

The presence of proprietary hardware-based network appliances, known as middleboxes, is a crucial part of the operation of today's computer and telecommunications networks. It supports a diverse set of network functions, such as firewalls, intrusion detection systems, load balancers, NAT, caches, and proxies [Martins et al. 2014]. For instance, in company networks, the number of these middleboxes is equivalent to the number of routers and switches deployed [Sherry et al. 2012].

However, the presence of these middleboxes brings several problems, such as [Han et al. 2015]: networks have become very complex with a wide variety of proprietary elements; the time to bring new services and functionalities to the market is high, as it depends on the production of new hardware; the operation of the networks is costly and depends on specialized knowledge in each proprietary platform; the costs of acquiring equipment to meet network demands are high, but they quickly reach obsolescence; lack of flexibility and scalability, as resources,

cannot be moved according to demands, and need to be scaled to the peak scenario; there are big barriers to innovation since it requires a significant investment to develop a device in hardware; the technologies of different manufacturers are incompatible with each other and do not allow reuse of hardware and software.

To address these problems and implement less costly and more flexible network infrastructure, NFV takes network functions of commercial appliances, actually dedicated hardware and software, to off-the-shelf equipment. By executing software-based functionality on commodity hardware with virtualization technologies for processing, storage, and networking, as exemplified in Figure 2.5, NFV frees a wide range of innovation solutions for new networks.

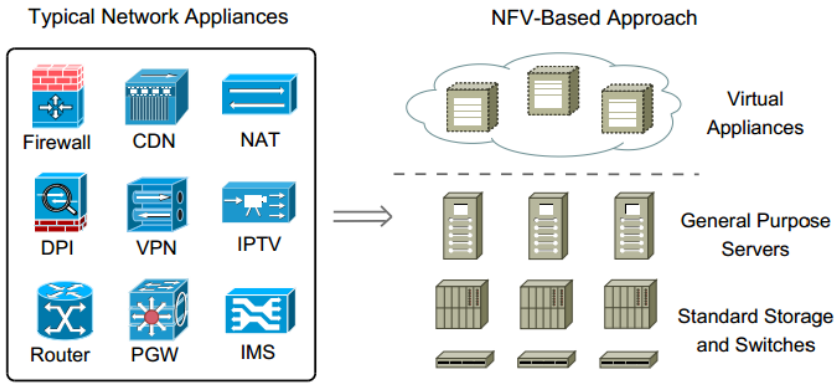


Figure 2.5: NFV implementation of network functions using virtualization techniques over standard hardware. Source: [Han et al. 2015].

The NFV objectives can be summarized as listed below [ETSI NFV ISG 2014]:

- Reduced CAPEX when compared to a specific target of hardware implementations: This goal is achieved by adopting commodity hardware and virtualization techniques, reducing the number of different hardware architectures, and resource sharing.
- Scalability and flexibility network functions: it is possible to decouple location from functionality and allocate network functions in the most appropriate places according to demands, increasing resiliency through virtualization, and making resource sharing easier. It becomes a strong groundbreaking mechanism on the next network generation as 5G networks.
- Software-based implementation: it incentives the innovation and time consuming for creating a new product in which reach to a specific market share.
- Reduced OPEX (operational expenses) by automating the procedures and making faster decisions for a specific event (e.g., fail, workload, and new demand).

- Reduced power consumption by migrating workloads and shutting down unused hardware.
- Interoperability through open and standardized interfaces between the network functions, the underlying infrastructure, and the associated management entities. In this way, the elements of NFV architecture can be implemented by different vendors.

These objectives have a huge impact on the business model of telecommunications networks, specifically in the new 5G networks. Consequently, this sector has received more investments in the NFV standardization, as discussed in the next section. However, it is essential to emphasize that the paradigm is applicable in both computer networks and telecommunications networks, especially at a time when both are converging to a resource-delivery model based on cloud computing.

2.2.1 Edge Computing

In addition to the increasing softwarization of infrastructure, content formats and production and consumption patterns are continuously changing as users demand improved Quality of Experience (QoE). This trend requires adaptation processes that respond to aspects such as personalization, localization, interactivity, and mobility. Therefore, edge computing brings a new paradigm in which shares computing, storage, and bandwidth resources as close as possible to the mobile devices or sensors. At the edge, the cloud services can deliver highly responsive time for mobile computing [Satyanarayanan 2017]. The nodes at the edge DCs may perform many computing tasks, such as data processing, caching, service delivery, and privacy protection.

ETSI defines the term MEC as a new platform that provides IT and cloud computing capabilities within the radio access network (RAN) near mobile subscribers [ETSI MEC ISG 2014]. The terms mobile edge computing and edge computing will be used in the context of the present work according to the scope. Those terms are viewed as necessary to enable specific use case classes defined, for instance, in 5G networks. The use cases have been classified into three service types (see, e.g. [1]): eMBB (enhanced Mobile BroadBand), URLLC (Ultra Reliability, and Low latency Communications), and mMTC (massive Machine Type Communications). A common characteristic of these use cases is the need for low E2E latency or provide a distributed content distribution. Therefore, we conclude that all call for some processing of data can be delineated by the proximity at the edge of the Radio Access Network (RAN).

As a result, we can foresee the emergence of this new kind of paradigm that presents vast differences from centralized cloud DCs. Once the edge DCs are

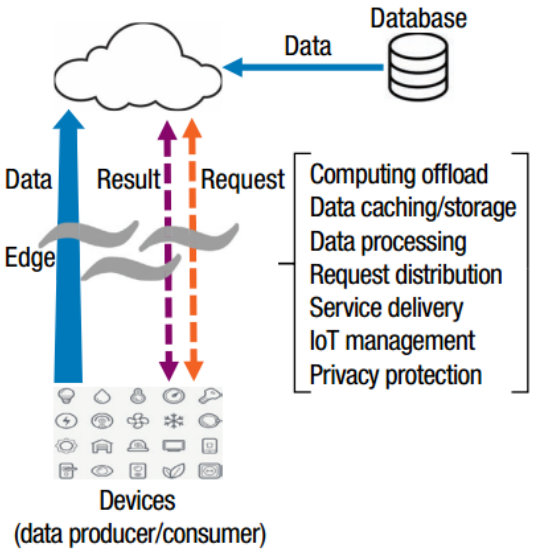


Figure 2.6: Edge computing paradigm. Source [Shi and Dustdar 2016].

geographically distributed, they have a much smaller scale in terms of resources when located close to the end-users; however, they support latency-critical applications [Mao et al. 2017]. To operationalize innovative services in MEC, the mobile network operators (MNOs) will need new technologies that enable the orchestration of multiple domains, which means across many distributed optical, wireless and cloud resources, as opposed to their traditional centralized model.

2.2.2 Multi-domain network orchestration

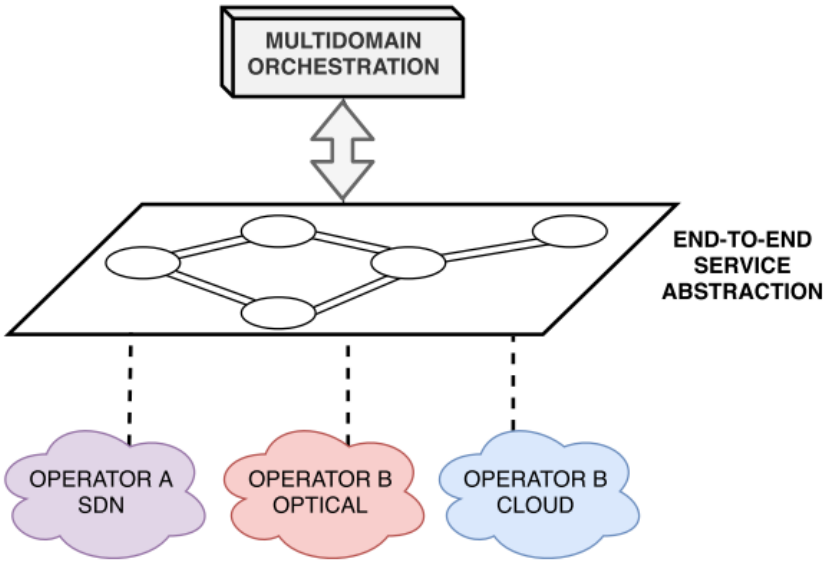


Figure 2.7: Context and scope of Network Service Orchestration (NSO). Source: [Saraiva de Sousa et al. 2019].

This section provides a brief background on SDN and NFV relationships with

multi-domain orchestration for next network generation. It will consist of infrastructure in which has heterogeneous specialized domains such as radio, access, transport, core, and (virtualized) data center networks. Deploying and operating E2E services are commonly manual and long processes performed via traditional Operation Support Systems (OSS) [Saraiva de Sousa et al. 2019].

By [Saraiva de Sousa et al. 2019], orchestration means a coordinated set of procedures to meet a given requirement (e.g., a customer requesting a specific network service). Beyond that, the orchestration also gives the capability of selecting and controlling multiple resources, services, and systems. From this manner, Figure 2.7 shows the context and scope of a called Network Service Orchestration (NSO) or a multi-domain orchestration entity, where it manages and controls the complete E2E services across different types of domains.

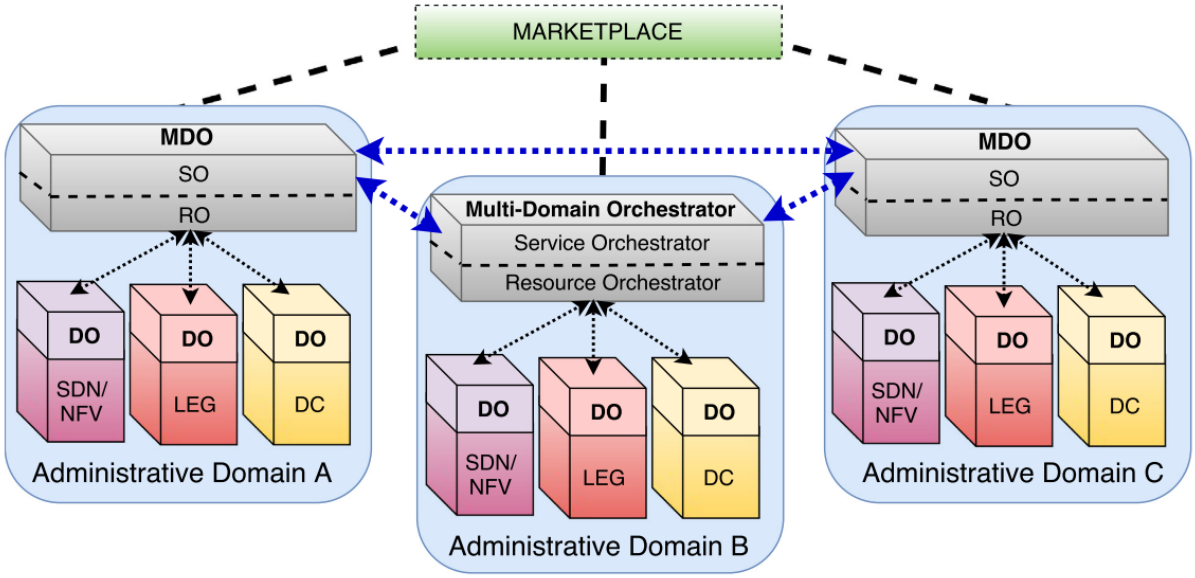


Figure 2.8: High-level reference model to illustrate the scope of Network Service Orchestration (NSO) in single-domain and multi-domain environment. Source: [Saraiva de Sousa et al. 2019].

2.2.3 Network Service Orchestration and standardization

Figure 2.8 shows how generic high-level reference model for multi-domain orchestration. Orchestration in the single and multi-domain environment is different. Once the orchestrator acts overall services and resource availability with its domains, as well as, it takes total control over those resources. A domain orchestrator manages the network service lifecycle and interacts with other components to control not only VNFs, but also computing, storage, and networking resources. The administrative boundaries of the provider limit its scope. As shown in Figure 2.8, domain orchestrators can comply with heterogeneous technological domains, such as SDN,

NFV, Legacy, and network resources like optical and wireless. Hence, multi-domain orchestration is more complex, since it is supposed to provide E2E services, which requires diverse resource interoperability [Rosa et al. 2015]. Currently, there is not a standard for the information exchange process in multi-domain environments, either multi-technology domains or multiple administrative domains.

2.3 Architecture of Next-Generation Mobile Networks (NGMN)

The last three decades have been marked by an exponential growth in the use of information technologies, with a society more and more dependent on the new technologies. For this, wireless network technologies have played a fundamental role in accessing services and applications almost anywhere, with user mobility as their main benefit. This development has been led by two strong technological trends, (i) Broadband Wireless Networks focused on the coverage of large areas for and (ii) Wireless Local Area Networks (WLAN) that have dominated for many years indoor wireless communications. In this chapter, a study of the main characteristics and technologies of the new network paradigm for wireless communications popularly known as Fifth Generation Networks is carried out. The relationship between Broadband Wireless and WiFi technologies is also studied. It analyzes the role of them in the future of wireless networks as well as the importance of coexistence and collaborative work. Lastly, it pushes out the development of a new generation of IEEE 802.11 standards to meet the demands and face challenges of wireless networks.

To address the socio-economic development of recent years, with the emergence of new business models and critical applications in different branches, the International Telecommunication Union (ITU) proposes the need for a new generation of mobile networks. The requirements that must be fulfilled by these networks began to be developed, preparing the open scenario for the investigation of new standards that would lead to the 5G paradigm [ITU-R Recommendation 2015].

2.3.0.1 5G network service requirements

The new 5G networks are characterized by a rapid response to allows multiple applications to provide several services simultaneously. It also seeks to enable the possibility of having completely reliable services anywhere and anytime with high quality performance independent of the access type to the system. These features are to be achieved even in high user density scenarios. The ITU-R IMT-2020 recommendation introduced the different service scenarios that the new mobile

networks must address, resulting in an expansion of existing IMT. Three main service cases were defined, becoming the main objectives to be achieved by the 5G networks, as shown in Figure 2.9.

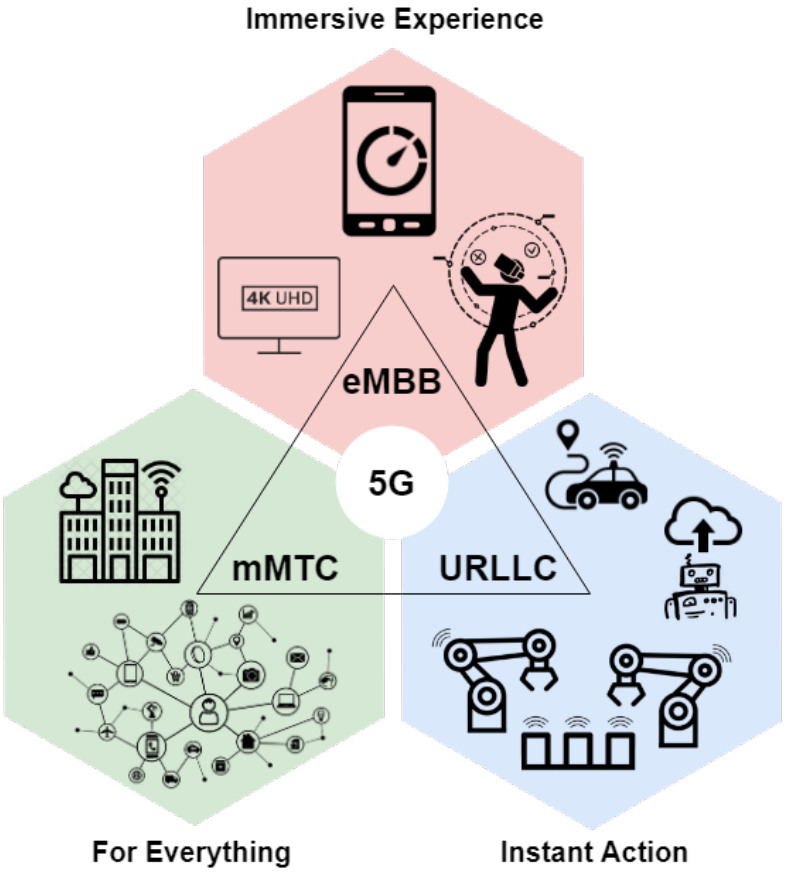


Figure 2.9: 5G Service Case: eMBB, mMTC and URLLC adapted from [Martínez 2019].

- Enhanced Mobile Broadband (eMBB): Mobile broadband addresses the human-centric use cases for access to multimedia content, services and data. A lot of attention have been devoted to this particular use case by the telecommunications industry since it directly represents the evolution of broadband mobile networks towards its fifth generation. For this, the main requirement that must be met in this type of scenario are the connections with very high peak data rates for all users. The objective of the eMBB service is to maximize data rates, so that values between 1 Gbps and 10 Gbps can be reached in real network scenarios.

- Ultra Reliable and Low Latency Communication (URLLC): This requirement is designed to address critical applications that present strict reliability and latency requirements, closely linked to the applications of Industry 4.0, future Healthcare and Vehicle-to-Anything communications (V2X) [Li et al. 2018]. In

these cases, the URLLC data transmission rates are relatively low, being the main objective to guarantee high reliability of data delivery with a Packet Error Rate (PER) less than 10^{-5}, which would represent networks with reliability of 99.99 %. In the transmissions of this type of applications, it is expected to achieve latencies for the control plane less than 10 ms and in the user plane less than 0.5 ms in the downlink or uplink communication in a separate way, representing a round trip time of only 1 ms, which means a reduction of 10 times the current latency in 4G networks.

- Massive Machine Type Communications (mMTC): 5G is not just about high-data rates but is an ecosystem of technologies that are going to provide a wide range of use cases and requirements. Basically, it also refers to IoT and about connecting devices without human intervention. It has been impacted by industries, businesses, and the lives of people. Therefore, it aims to provide connectivity to a massive number of devices whose traffic profile is typically a small amount of data (spread) sporadically. In this case, latency and throughput are not a big concern. However, the main concern is about power utilization optimization on those devices. Such devices expect to safe battery life as much as possible to achieve around ten years.

Simultaneously addressing all these use cases for a wide range of applications, it is stated that the 5G network paradigm provides access to services $4A$, which means Access Anytime, Anywhere, Anyone and Anything. We are in the presence of a flexible and intelligent network architecture, capable of executing Tasks through a virtualized platform and software defined orchestration. This architecture, based on network programmability, will be able to analyze all network data in real time to customize the services availability and the resources allocation. The new 5G network is differentiated from 4G networks by the evolution in performance at the radio level and by providing a great E2E flexibility, contributed by the introduction of a network softwarization approach. With these features operating in an integrated way, is projected the 5G network as a complex service-oriented network architecture [Yousaf et al. 2017].

2.3.1 Network Programmability for Expressive Orchestration

The 5G architecture is expected to be network-agnostic, where the network core will be common for all RATs that will be supported by the network, existing radio interfaces, and the New Radio (NR) interface introduce for 5G. This feature makes it necessary to implement rigorous control mechanisms that allow uncoupling the network core from access technologies. At the same time, other mechanisms should

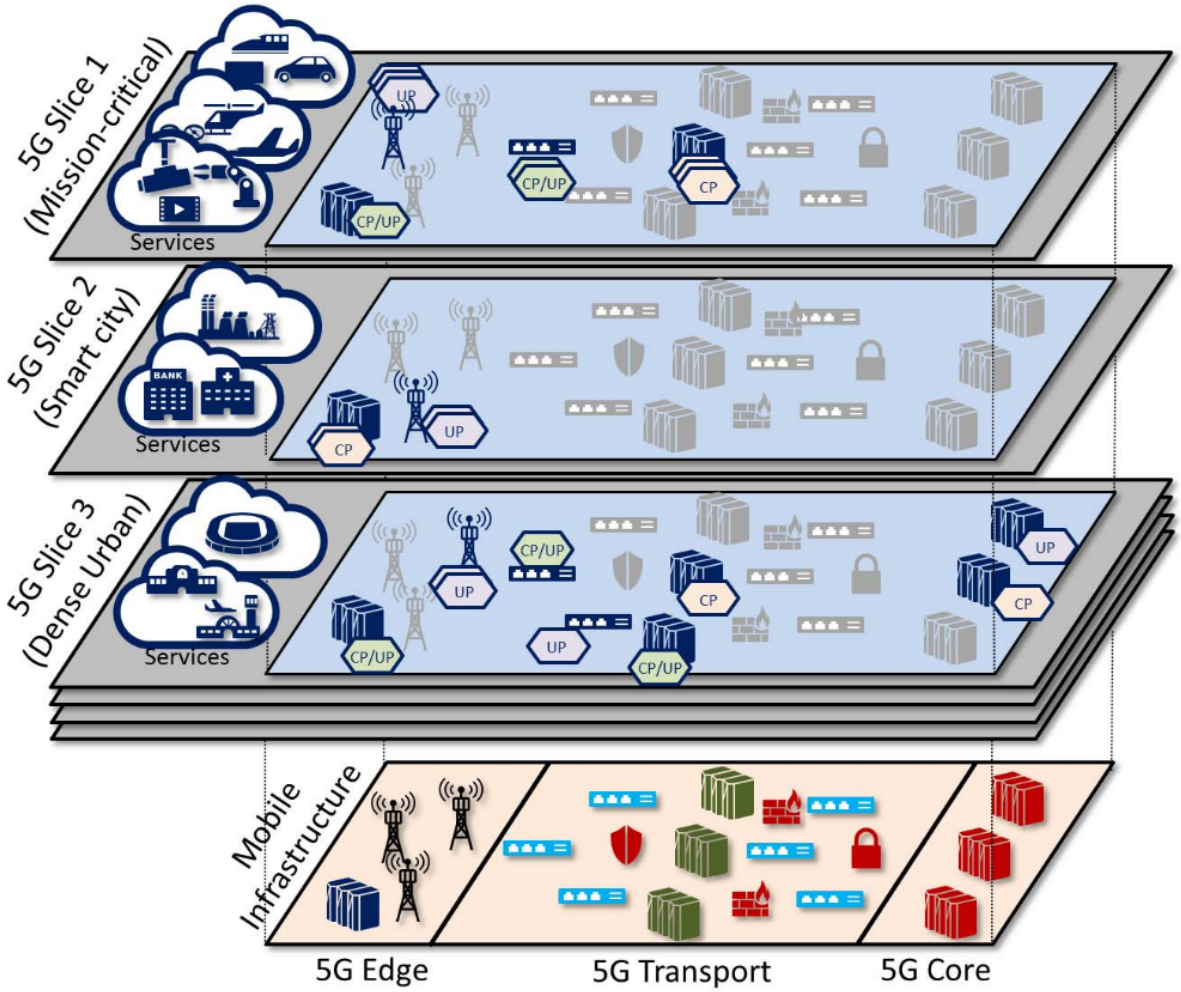


Figure 2.10: 5G network slicing [Yousaf et al. 2017].

be implemented to orchestrate the inter-working of this multi-RAT, 5G NR access network, mobile access networks such as LTE-A, WLAN, and fixed networks. These mechanisms must be capable of mobility management and user session management for the continuity of the services during the transition from one technology to another. The multi-connectivity offered by the access through multi-RAT provides robustness to the network and better throughput performance.

With the network resources virtualization, multiple networks slices with different characteristics can be implemented to address different service cases, as shown in Figure 2.10, maximizing the network performance, QoS and Quality of Experience (QoE). The implementation of the network slicing is one of the main techniques in the deployment of 5G architectures [Li et al. 2017]. They allow network operators to implement virtual logical networks and sets of network functions for different services over the same physical infrastructure. As each network slice is a logical entity independent of each other, it allows independent and customized functions of Management and Orchestration (MO) for the services. Virtualization also offers the

possibility of having a network architecture with distributed functions that actively contributes to the reduction of core and backhaul traffic by placing many services on edge networks closer to users. Edge computing is assuredly beneficial in terms of latency for many critical applications in URLLC.

The orchestration and management of the network should, therefore, allow maximum performance during the implementation of distributed functions, network slicing, and resource allocation. With different network domains working under the range of 5G access technologies [Baranda et al. 2018], unified network management is one of the biggest challenges in these architectures to ensure compatibility and flexibility among all technologies. Within these scenarios, the management of network slices to implement solutions of various service cases becomes the heart of these architectures. The management must provide functionality such as optimization and capacity planning for a slice, granting the necessary resources according to the service type requested. It must be able to monitor the quality of the provisioned slice to take the necessary actions in case of re-optimization. Other vital functions that must be fulfilled are the management of the slice fault, inter-slice orchestration, management of slice security and monitoring, and analysis of slices resources.

2.4 Related Works

This section describes the related works focusing in network programmability for efficient orchestration. Firstly, in Subsection 2.4.1 we position the capabilities of orchestration across optical, wireless and cloud domains, showing the limitations of current solutions. Furthermore, Subsection 2.4.2, we compare our solutions with related works in the areas of seamless reliable mobility for convergent networks.

2.4.1 Orchestration across heterogeneous resources

The heterogeneity of optical, wireless and cloud networks is requiring a high level of virtualization to enable the hierarchical decoupling of the software from the underlying hardware by abstracting physical resources as shown in Figure 2.11. It poses a new set of challenges that require the joint control of optical, wireless and cloud environments [Boulogeorgos et al. 2018]. The resources generally sit across different domains: time, frequency, and space in wireless environments; optical fibers, wavelengths, and switches/lasers in wired environments [Fu et al. 2018]; placement, and computing resources in cloud environments [Bogue 2017].

As mentioned by [Nencioni et al. 2018], the possible future research challenges denoted in Figure 2.12, whereas is worth noting, softwarization technologies as NFV

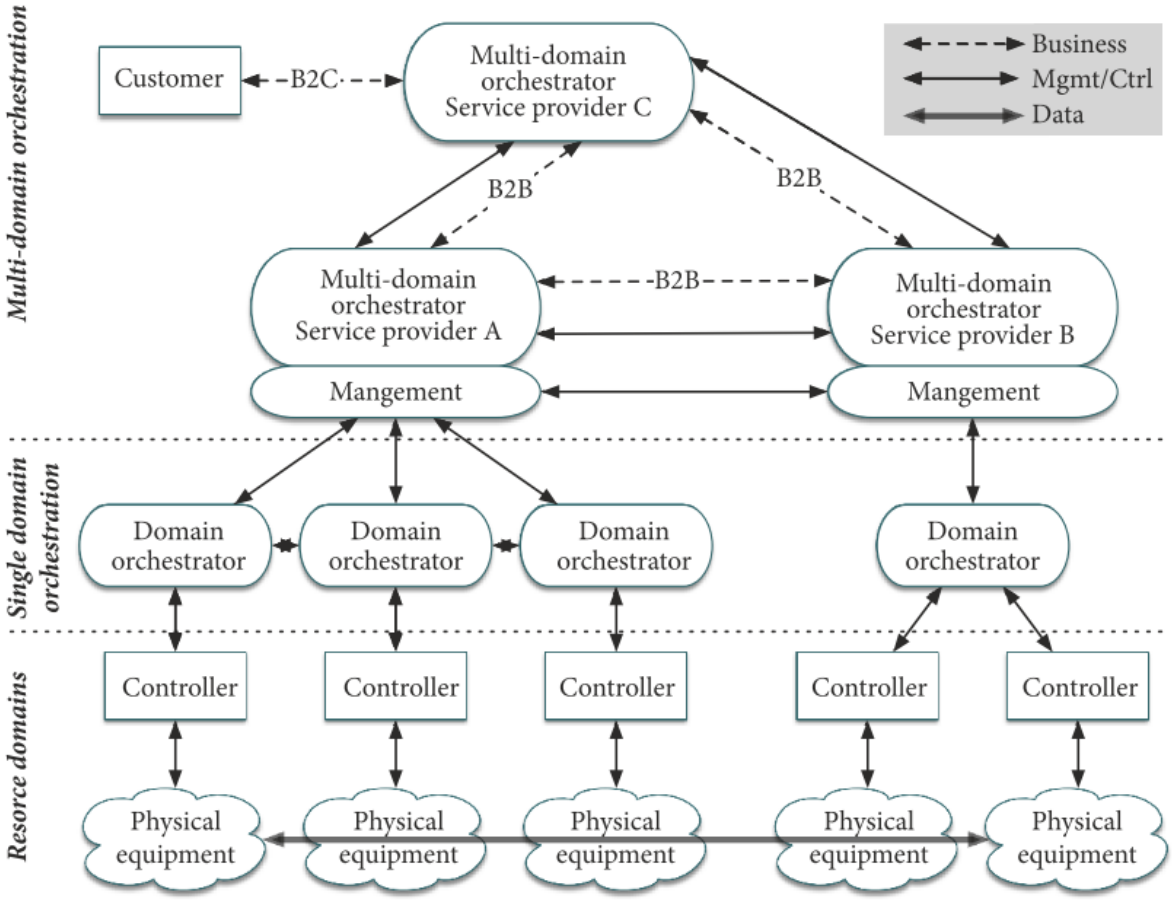


Figure 2.11: Cross-layer E2E Control and Orchestration [Guerzoni et al. 2017].

and SDN are the main enablers to develop novel solutions on top of new network generation (e.g., 5G network). The centralized coordinator and control increases flexibility and agility in service creating, and provides a customized network services for each customer (i.e., slice). the success of the network virtualization heavily relies on orchestrating end controlling capabilities, where optimizes the capital expenditure and operating costs (CAPEX and OPEX). Additionally, the softwarization technologies help to adapt the network in responding of aspects such as personalization, localization, interactivity, and mobility.

Table 2.1: Research projects for networking experimentation [Both et al. 2019].

Project	Objective	Main Tools		Orchestration and Programmability	
		Manage	Slicing	Physical Layer	Link Layer
GENI: Global Environment for Network Innovations	Provide a virtual laboratory for networking	GCF	FlowVisor; OpenVirteX FlowSpace Firewall	Optical wavelengths	Switch packet; Flows Handover
OFELIA: Open Flow in Europe - Linking Infrastructure and Applications	OpenFlow-based to control the network environment	OCF	Optical FlowVisor VerTIGO	Optical wavelengths	Optical ports; Flows
OF@TEIN: OpenFlow/OpenFederation at Trans-Eurasia Information Network	OpenFlow-based SDN	OCF	FlowVisor; VLAN-based	Optical switching	Switch packet; Flows
RISE: Research Infrastructure for large-Scale Experiments	OpenFlow infrastructure	RISE Orchestrator	MAC Rewriting	None	Switch Port & Packet; Flows
FIBRE: Future Internet Brazilian Environment for Experimentation	Future Internet testbed	OMF6	FlowVisor; TDD	Wireless channels	Switch Port & Packet; Flows
FUTEBOL: Federated Union of Telecommunications Research Facilities for an EU-Brazil Open Laboratory	Optical, wireless, and cloud convergence	CBTM; FOAM; O2CMF	TDD; Optical wavelengths Virtual Infrastructure	Optical-Wireless Channels and Bandwidth	Switch Port & Packet; Flows

For this purpose, there is a significant number of projects that have developed experimental facilities for networking experimentation in which open the creation of innovation and the new groundbreaking applications. In the United States, GENI established an open infrastructure for at-scale networking and distributed systems research and education [Huang et al. 2017]. In Europe, OFELIA created an experimental facility that allows researchers to control the network precisely and dynamically uses OpenFlow to control the network environment [Huang et al. 2017]. The OF@TEIN project was designed to deploy a shared OpenFlow-based SDN testbed infrastructure [Huang et al. 2017]. Similarly, Japan has the RISE project, which offers OpenFlow-based research infrastructure on the Japan Gigabit Network [Huang et al. 2017]. In Brazil, the FIBRE project built federated testbeds for research and education. FUTEBOL was conceived to enable experimental research on optical, wireless, and cloud convergence [Marques et al. 2017].

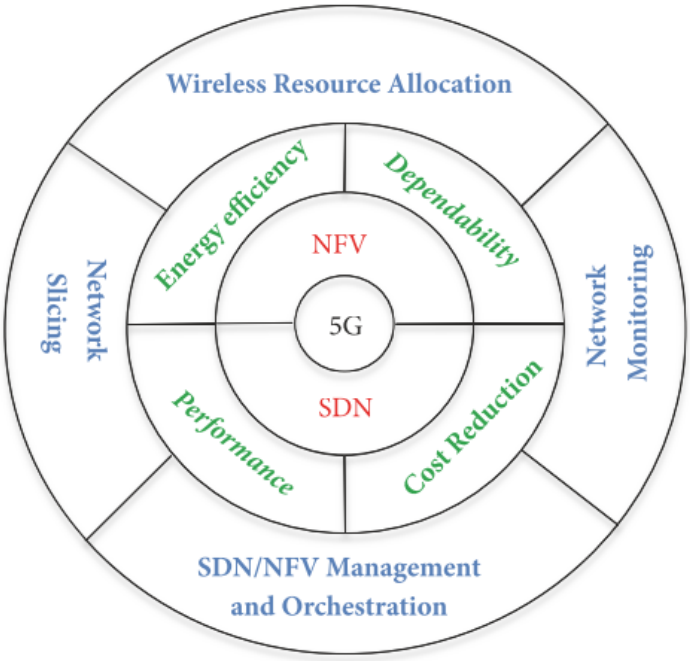


Figure 2.12: Sketch of possible future research challenges: in red, the softwarization technologies; in green, the objectives; in blue, the tasks. [Nencioni et al. 2018].

Experimental infrastructure should provide flexible configuration through orchestration and programmability. The analysis summarized in Table 2.1 reflects that FUTEBOL differs from all other projects through its integrated orchestration of wireless, optical, packet, and cloud network resources [Both et al. 2019]. Besides, FUTEBOL extends the range of configurable physical layer (*e.g.*, wireless and optical channels with adaptive bandwidth) and link layer (*e.g.*, switch ports and packet, flows, and wireless handover) parameters that are exposed to network controllers.

In this manner, it allows multi-domain programmability[5] for the experimenters interested in controlling physical and link layer parameters from different domains.

2.4.2 Orchestration for seamless and reliable mobility

SDN, a networking paradigm that separates control functions from the data plane, is being enhanced into software-defined wireless networks (SDWN) [Fontes et al. 2017] by adding clients' seamless mobility and a better quality of service. SDWN makes it possible to create handover mechanisms in which migration decisions are taken by SDN controllers instead of STAs and allows more effectively to update the locations of the clients in the network and the related backhaul routes. In 802.11 networks working in infrastructure mode, the STAs are associated with an AP, which can serve several clients while acting as a bridge to access the services of the network [Mishra et al. 2003]. As a consequence of the mobility offered by these networks, the STAs need to associate with different APs to maintain connectivity when entering a different coverage area. This mechanism is driven by the clients and not by the network infrastructure. It is based on several criteria used by manufacturers, such as the loss of beacon frames or the degradation of link quality [Raghavendra et al. 2007].

Several studies have been conducted to implement mobility management in SDWN. Ethanol, an SDN architecture for WiFi networks, is proposed in [Moura et al. 2015]. This solution can control customer mobility characteristics and association/disassociation processes by denying association requests from the STA, forcing it to associate with another AP. SDWN is proposed in [Suresh et al. 2012, Gilani et al. 2017], where the creation of virtual AP for each client allows a seamless handoff process through the migration of the virtual AP that serves the user who needs to move from physical AP. In these solutions, although handoff processes are guaranteed without significant degradation in throughput, the aspects of updating backhaul after migrations are not addressed.

A mobility management mechanism presented in [Sanchez et al. 2016]. The work's solution is based on the concept domains and on updates made in the backhaul, depending on whether migrations are intra-domain or inter-domain. Though, it does not analyze the delays that the exchange of management frames during the re-association process introduces. In [Zehl et al. 2016], a solution called BIGAP is proposed, which proposes that all APs are configured with the same BSSID, creating a single global BSSID, but with different channel configuration on each AP. When the handoff decision has to be made, the SDN controller copies the client's association

[5] Multi-domain in this thesis is not an inter-domain network of autonomous systems, but it is defined as a heterogeneous network technology composed by wireless, optical or packet domains.

information to the destination AP and sends out a beacon containing a Channel Switch Announcement (CSA) with the channel set to the target AP. After the destination AP is known, the SDN controller also updates the ARP tables or the routes in the backhaul switches. Although BIGAP efficiently addresses migration's issue to different APs, and re-association delays (i.e., updating the backhaul switches), its implementation becomes complex due to the number of states that has to manage. The synchronization that is needed for the migration and for updating routes.

These proposals majority have one or more controllers triggering the handoff process and performing client's association updates to a given target AP. However, none of them analyzes the occurrence of failures in the APs. This phenomenon would cause a classic handover process once the controllers could not migrate the client information for the available APs. Besides, these studies have maintained the operation in classic infrastructure mode of the WiFi networks. Unavoidably, this type of operation forces clients during their mobility to re-associate with new APs, which imposes the need to find solutions that minimize re-association delays issue. In [Gowda et al. 2016], the mobility is placed in an AP by using a robot to obtain the best SNR between APs and STAs. Thus, if the normal 802.11 infrastructure mode is split into functional blocks, mobility could be placed in the AP, and the access to network services could happen through non-mobile STAs. In this context, multi-connectivity can be guaranteed with the mobile terminal even if it has just one wireless interface, and allows to have more efficient handoff processes and failover mechanism.

Re-transmission schemes are one of the most used techniques to increase the reliability and availability of communication systems [Buccheri et al. 2018], but such techniques introduce latency into the system. On the other hand, diversity schemes are today considered fundamental to achieve reliable communications in wireless channels [Pocovi et al. 2018]. We can have different diversity methods, such as frequency diversity (e.g., by using Orthogonal Frequency Division Multiplexing (OFDM) systems to combat frequency fading of multipath wireless channels and spatial diversity) [Popovski et al. 2018].

Regarding the diversity, Parallel Redundancy Protocol (PRP) concepts are implemented in [Rentschler and Laukemann 2012], where uses a diversity method to operate in IEEE 802.11 networks through parallel wireless links. The motivation behind it is that two sufficiently uncorrelated channels can provide more reliable communication with reduced latency compared to the transmission over a single wireless channel. Overall, the average latency considering the parallel redundancy path is significantly better than the one experienced by each link independently [Cena et al. 2018]. The use of PRP wireless infrastructures for industrial control applications is formalized in [Yang and Cheng 2019] through worst-case delays anal-

ysis by using network calculus theory. To evaluate the advantages of redundant WiFi communications, [Lucas-Estañ et al. 2018] performs one of the first studies in real scenarios with mobile industrial applications. However, in energy-constrained applications (e.g., cloud robotics and IoT) by using multiple interfaces directly impacts battery usage, limiting the autonomy of mobile devices.

Software-Defined Wireless Networks (SDWN) solutions can help with client mobility and QoS demand through network-centric schemes with global views of the network to reduce the mobility interruption time (MIT) [Gilani et al. 2017, Zeljković et al. 2018]. Several proposed frameworks use SDN orchestration together with prediction mechanisms to perform more efficient handovers [Liu et al. 2019, Zeljković et al. 2019]. Many works base their solutions on virtual AP (VAP) as a virtual network function (VNF) for each client. High communication performance during handover processes is reported despite the absence of fault recovery mechanisms. The use of SDWN architectures allows for the use of multiple connection schemes with mobile elements to achieve more reliable communications and fault recovery capabilities. A multipath transmission architecture is proposed in [Xu et al. 2017], which creates a connection with a VAP through multiple physical APs. In [Martínez et al. 2018], an SDWN architecture explores multi-connectivity and achieves improved communication reliability only with one radio interface on the mobile element. Finally, only a few works address diversity for redundant transmission over several wireless links (e.g., [Nielsen et al. 2018]).

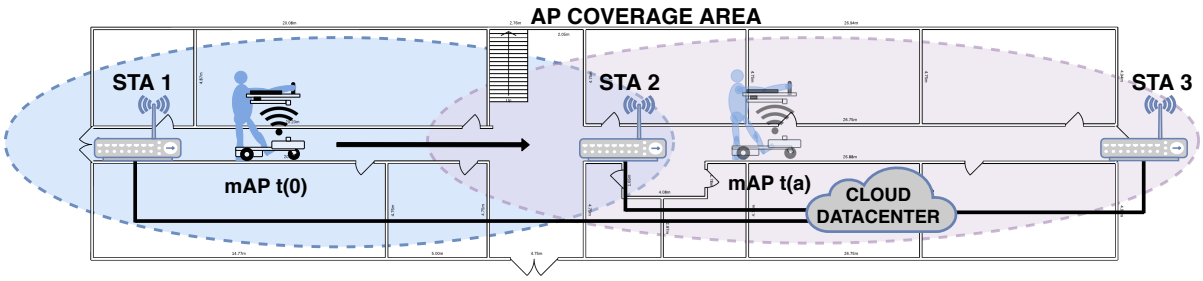


Figure 2.13: Experiment Scenario: Infrastructure [Martínez et al. 2018].

The SDWN aimed to achieve more reliable communication when mobile robots are deployed, for example, in applications involving robotic devices that assist impaired individuals locomotion [Martínez et al. 2018]. The architecture splits the functions in 802.11 network infrastructure, creating a mobile AP (mAP) located in the robot and an access network composed by non-mobile STAs that communicate with the cloud, as presented in Figure 2.13 about the experiment scenario. A centralized SDN controller is in charge of mobility management, operating according to the information of the physical environment provided by the local agents installed in the STAs. In this context, multi-connectivity can be guaranteed even if the STA has

just one wireless interface, thus allowing efficient handover processes and failover mechanisms to reach ultra-reliable communications.

2.4.3 Agile orchestration by using a source-routing mechanism

In several source-routing schemes, a traditional switching SSR is represented by outgoing ports, where the routeID ships a stack of switches and ports to send across the whole path. ELMO and SecondNet are works that explore port switching SSR. Recent works [Liberato et al. 2018, Dominicini et al. 2020] has brought RNS-based SSR as an alternative method to perform SSR that defines the outgoing port on each node by using modulo operation between the RouteID and the nodeID. This proposal of thesis brings evidence that RNS-based SSR can be explored by using different polynomial based in the residues system that would give more expressiveness for an E2E communication in convergent networks, which, indeed, would open the creation of new groundbreaking applications. Hence, the proposal of the thesis presents many exciting properties that not exists in the RNS-based SSR, such as:

- Multipath packet forwarding: The forwarding operation is the direct result of the modulo operation between routeID and nodeID. In this case, we have another polynomial as a result of this operation that means the outgoing port.
- Multi resource reconfiguration embedded in the same path: As explained by the works [Martinello et al. 2014, Gomes et al. 2016], one of the principles of the source-routing is the fact that the order does not impact the forwarding on each node. By using the modulo operation each node extract its output information. Therefore, it is relevant to note that embracing information from different resources pushes ahead the expressiveness to the forwarding capabilities, in which each layer would be able to actuate in the result of the modulo operation based on its behavior.

Therefore, we can explore these properties to provide networking functionalities that cannot be covered by previous works. The following topics exemplify some potential applications of these properties:

- Multipath packet forwarding: This property allows embedding in the routeID the delivering of a packet to different paths in a given hop. For instance, KAR [Gomes et al. 2016] is a fast failure reaction scheme that uses RNS-based SSR and explores this property. It proactively adds redundant nodes in the routeID to create resilient forwarding paths (called protection paths). ELMO [Shahbaz et al. 2019], BIER [Wijnands et al. 2017], and [Reed et al. 2016] use

a source-routing scheme to addresses the multicast scalability problem in multi-tenant data-centers and to overcome the group scalability presented in the legacy protocol, such as [Cain et al. 2002, Fenner et al. 2016].

- Multi-domain information embedded in the same path: It is essential to note that the order of the forwarding information in the path is irrelevant: This property allows different embedding domains or redundant information from each node in the routeID, even if they come from different domains. They are disjointed of the desired route. This property is enabling scenarios where header modification is challenging to implement, likewise, need to implement novel forwarding method, such as wireless and optical switches [Wessing et al. 2002].

Initially, RNS-based SSR applied to optical packet-switched networks was explored by [Wessing et al. 2002] to avoid header rewriting and label distribution protocols. Further, KeyFlow [Martinello et al. 2014] integrated with SDN in core packet-switched networks, which builds a fabric model that replaces the table lookup in the forwarding engine for RNS operations. Then, [Shahbaz et al. 2019, Gomes et al. 2016] works explored additional properties of RNS to extend KeyFlow. [Shahbaz et al. 2019] explored techniques to reduce the length of the forwarding label, while KAR deviates packets from faulty links with routing deflections and guides them to their original destination due to resilient paths added to the forwarding label. Beyond these ideas, PolKA has also brought an RNS encoding based on polynomial that has given programmable, expressive, scalable, and agile Service Function Chaining for Edge Data Centers.

As a previous work for multipath forwarding was [Reed et al. 2016] that uses source-routing following Bloom Filter approach, implemented natively in SDN, as a new technique for providing stateless multicast switching at operator scale. The problem in using Bloom Filter is that the routeID has to grow to maintain a desired false-positive probability. Moreover, their theoretical proposal presents some critical limitations for real-world scenarios: (i) in contrast to our approach that performs one modulo operation per core node, their data plane has increases considerably the number of rules and tables. The work creates tables for each port to achieve the pipelining transmission that supports multicast forwarding (i.e., to all outgoing ports obtained from the routeID). It also implies scalability and performance issues because there is a limited number of tables and TCAM entries supported by the OpenFlow switches; ii) they reused the IPv4 and IPv6 headers to ship the routeID, therefore, breaking the compatibility in the whole infrastructure by creating a clean-slate solution. Their IPv4 and IPv6 packet headers do not have standard behavior.

From a different perspective, [Shahbaz et al. 2019] proposed an optimized encod-

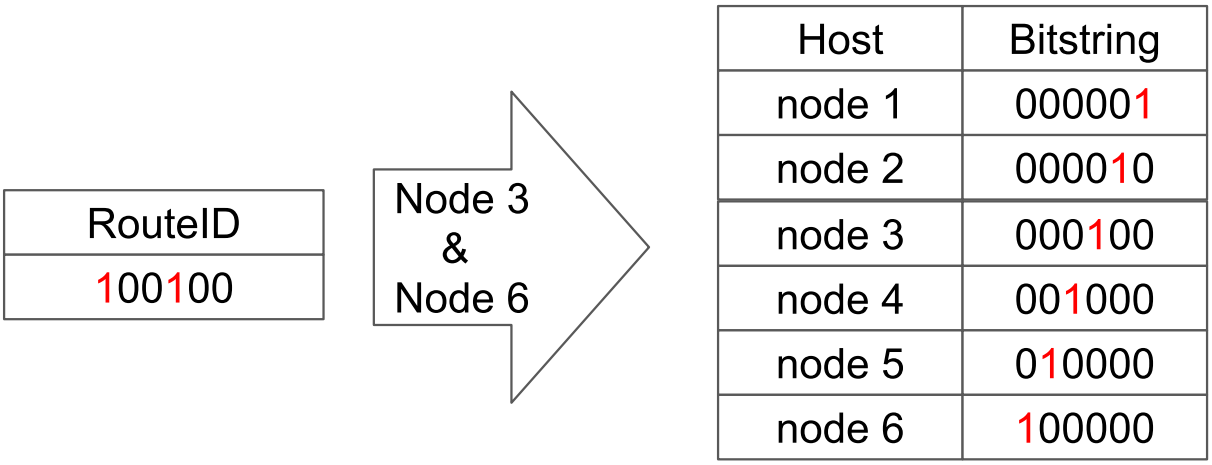


Figure 2.14: BIER routeID composition.

ing scheme by using bitmaps written for $P4_{14}$ Mellanox hardware switches; however, considering a different overview, it is not a topology-agnostic solution that implies a considerable barrier to develop novel solutions. Differently, BIER described in RFC 8279 [Wijnands et al. 2017] has proposed a hybrid approach, where there is a table on each switch previously set defining the next-hop (i.e., it can be set based on some IGP protocol such as OSPF or IS-IS), and the routeID in which defines the list of switches where that packet has to be steered, as shown by Figure 2.14. Depending on the hop, the data plane must perform the and operation, modify the routeID header, and the cloning operation to one or many interfaces for a given packet, according to the Figure 2.15. To fast explore different paths, the controller and orchestration layer has to set all the table's states to act in some event of changing in the topology (e.g., failure, traffic engineering). The time in reacting to some failure or the granularity to fast express different becomes a stringent obstacle in delivering new modern applications, especially those that need ultra-reliability and low latency.

To this end, we envision M-PolKA as a generalized RNS-based SSR by using different polynomial bases and schemes to provide a reliable routing across heterogeneous resources, such as wireless and optical. This mechanism explores the Chinese Remainder Theorem in finite fields of two or more elements [Schroeder 2009]. It has been a shift from integer to polynomial arithmetic proposed by [Dominicini et al. 2020] that enables performance optimization and reuse of off-the-shelf network hardware, such as CRC (cyclic redundancy check). We also explore the use of CRC Polynomial to be used in off-the-shelf hardware and high-performance switches (i.e., with the support for 10Gbps, 40Gbps, 100Gpbs of throughput.) while portending new network services.

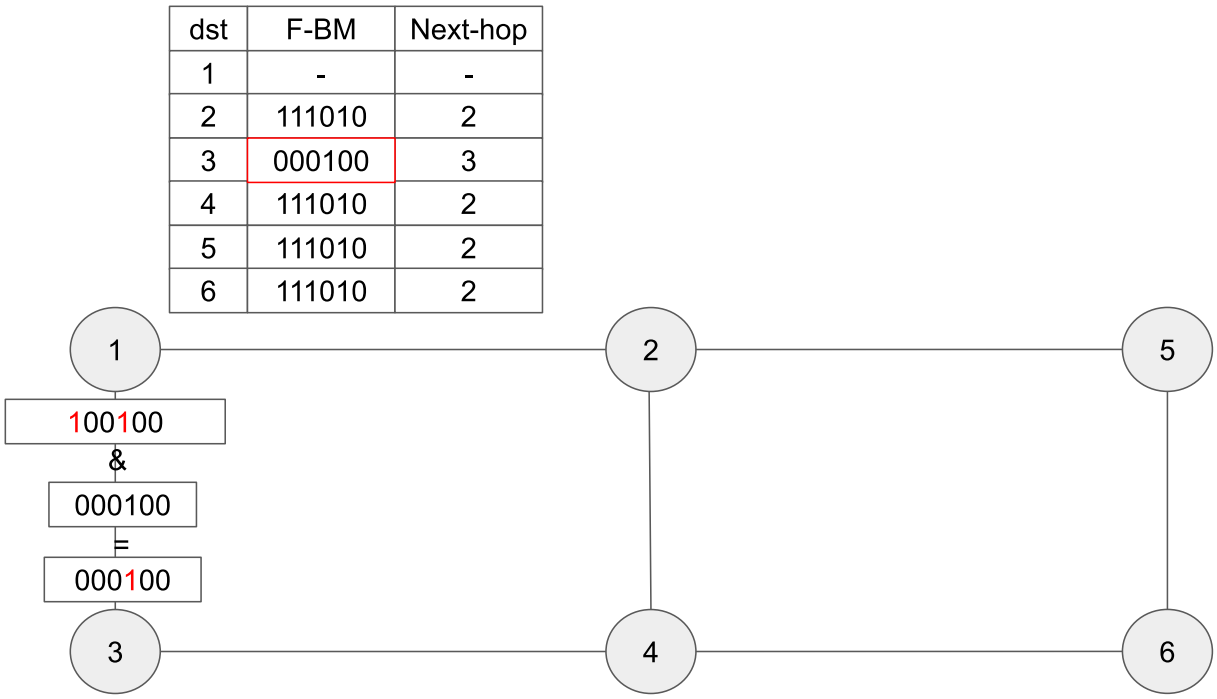


Figure 2.15: Example of BIER table at the Node 1 based on a topology with 6 nodes. Adapted from [Merling et al. 2018]

2.5 Chapter Remarks

In this Thesis, we advocate that the most appropriate expressive orchestration and routing methods offer an agile and flexible handover solution for reliable applications in a high-mobility scenario. A source routing algorithmic routing computation mechanism becomes crucial by eliminating the lookup table time on each switch, which gives more determinism in E2E communication. In this way, we eliminate tables in intermediary core nodes to reduce latency, jitter, and control plane signaling. The path information is inserted in a packet header only once when the packets traverse ingress the edge nodes. Then, each edge core node can make forwarding decisions based on a simple operation over a field of the packet header. Regarding the hop count, as a logically centralized SDN Controller calculates the path and decides to choose the minimal routing or not, depending on traffic engineering aspects.

The primary advantage of source-routing is the elimination of tables and complex routing responsibilities from intermediate nodes, placing the responsibility for route selection at the ingress nodes [Sunshine 1977]. Some source routing methods accomplish this by representing the path to any destination as a list of segment addresses [Filsfils et al. 2015] or port labels [Soliman et al. 2012] traversed to reach that destination. Then, the list is embedded at the ingress node in a packet header, and used by each node in the path to take its forwarding decision.

Chapter 3

Enabling heterogeneous resource orchestration and cross-layer programmability

3.1 Overview

Emerging telecommunication trends, such as cell densification, millimeter-wave, heterogeneous networks, and E2E communications, pose stringent throughput, latency, energy, and cost requirements that demand the coordinated control and management of heterogeneous resources as optical, wireless and cloud [Boulogeorgos et al. 2018]. In this way, the design of next-generation networks must offer an orchestration architecture to virtualize and manage heterogeneous resources across different network infrastructures in a unified manner. Several controllers offer northbound interfaces that can be exploited to put into operation orchestration tasks.

Notwithstanding, such orchestrator must control heterogeneous network resources that deal with the different aspects and belongs to each layer: time, frequency, wavelength, and placement [Fu et al. 2018]. Besides, the heterogeneity of optical and wireless networks requires a high level of virtualization to decouple the software from the underlying hardware, by abstracting physical resources. Despite all innovative features brought by SDN architecture, there is still, however, a barrier in dealing with the physical layer to foster the programmability of physical parameters, once the SDN architecture was not conceived to deal with it. By promoting functional cross-layer control and management over heterogeneous resources, we can deliver programmability for a given E2E communication. Hence, the orchestrator can provide better E2E communication by ranging resources from physical to application layers. There is a need to meet the diverse E2E requirements that are being fulfilled adequately by the networks [Yousaf et al. 2017].

This chapter describes the design principles, the architecture, and evaluation of our orchestration architecture, which was integrated into the FUTEBOL Control Framework (FUTEBOL CF) [Both et al. 2019]. Jointly with our orchestration architecture, FUTEBOL CF can provide a unified optical-wireless-cloud orchestration, including: (*i*) cloud and NFV orchestration capabilities, such as policy-based automatic scaling of a VNF, resource monitoring across testbeds based on container virtualization, and VNF life-cycle management; and (*ii*) cross-layer programmability by interacting with the multiple network controllers in the wireless and optical domains that virtualize physical network resources.

As experiment validation, we separate in two use-cases:

- The use-case 1, Section 3.3.2, we experimentally demonstrate our orchestration architecture of LTE multi-cell resource allocation in an integrated LTE-over-PON architecture. The orchestrator dynamically adjusts the optical and wireless bandwidth of each cell, according to their demand, jointly with their fronthaul rate and reserved PON capacity onto an inter-testbed cloud resource.

- The use-case 2, Section 3.4, we analyze the impact of scaling physical resources, optical and wireless, according to mobile users' demand on LTE/WiFi networks. Finally, we analyze the effects of dynamically allocating computing resources (vertical scaling) using a cloud auto-scaling feature, which adjusts the available resources according to the demands from a mobile application to a web service.

Finally, our results show the impact of cross-layer programmability in the orchestration for better E2E service performance, demonstrating the benefits of our orchestration architecture shipped as part of the FUTEBOL CF.

3.2 Proposal

This section describes the design principles, architecture and enablers, and the potential limitations of our following proposals: an orchestration across heterogeneous resources, enabling cross-layer programmability.

3.2.1 Design principles

The design principles of our proposal architecture were motivated by the orchestration challenge on heterogeneous resources that can enable cross-layer programmability — it supports innovation, expressive and agile orchestration, fostering the infrastructure slicing optimization.

To address the early-mentioned challenges, we design the solution based on the following enablers:

- **Introduction of a cross-layer network programmability model**: By using Openflow SDN switches, we can program the forwarding mechanism for a given service, in a centralized manner. However, the current SDN approaches were not designed to deal with physical programmability. Therefore, by leveraging the programmability of physical parameters (e.g., the wavelength of optical switches, FPGAs, and wireless), we expand the orchestration observability, which gives a more refined control of the resources needed for a given E2E communication.

- **Unified orchestration control with management interfaces**: this enabler is introduced to design a unifying API for South Bound Interface (SBI) to provide underlying hardware management. For the North Bound Interface (NBI), it is aimed to facilitate the routing, control, and management across heterogeneous resources. Also, a unified orchestration means that the orchestrator is topology-aware which makes better decisions based on server monitoring information and knowledge of the topology. It optimizes E2E provisioning and reconfiguration, since the resources are distributed through different places (i.e., inter- and intra-testbeds communications). For example, the orchestrator can place a given service application in the shortest path between source and destination or spread service applications over different domains to load-balance the demand.

3.2.2 Architecture design and enablers

Figure 3.1 shows a big picture of FUTEBOL CF architecture. We highlight the components that we have incorporated to provide integrated control of optical, wireless, and cloud resources through different testbeds. It allows the advancement of experimental research telecommunication, which comprises heterogeneous resources from distributed testbeds. The separation of functionalities and components distinguishes among: (*i*) Physical Infrastructure Layer (optical, wireless, and cloud resources); (*ii*) Virtualization Layer; (*iii*) Experiment Control and Orchestration Layer; (*iv*) Service Layer; and (*v*) Testbed Management Layer.

The Application-Based Network Orchestration (ABNO), shown in Figure 3.1, is based on IETF RFC 7941 and works as a network orchestrator of multiple resources in the FUTEBOL CF. Initially, ABNO was in charge of the E2E control of optical and packet network resources at the University of Bristol (UNIVIBRIS). However, the orchestration for E2E network control and management essentially needs to be

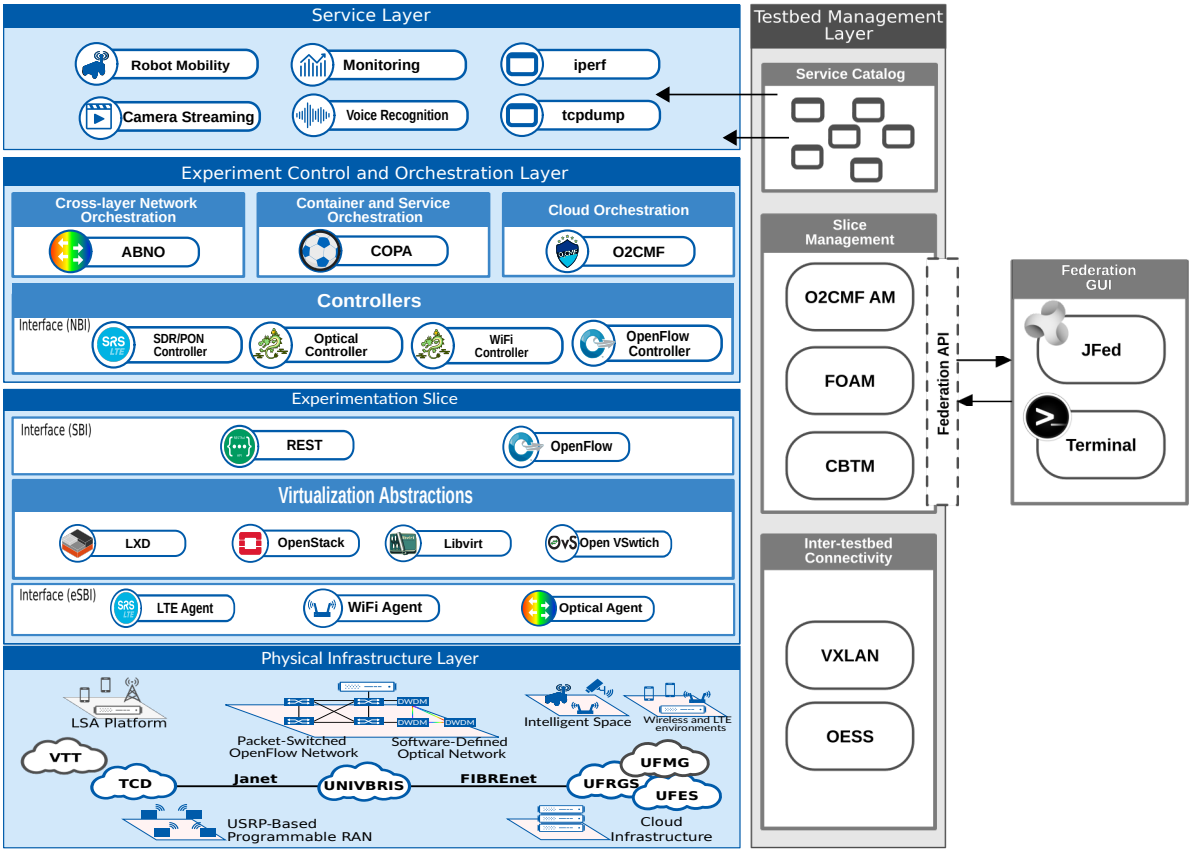


Figure 3.1: Orchestration of heterogeneous resources in an intra-domain architecture enabling cross-layer network programmability. Adapted from [Both et al. 2019]

improved, especially when virtual network operators come into the next-generation mobile networks. Most of the existing SDN controllers rely on monolithic software, by offering insufficient flexibility for heterogeneous resource controlling.

Our focus lies on providing a programmable forwarding mechanism and physical parameters for a unified control and management architecture and topology-aware orchestration. For this purpose, we have extended ABNO and the underlying control to perform orchestration with wireless, optical, and OpenFlow networks, where OpenStack and OpenFlow Control Management Framework (O2CMF) [Ceravolo et al. 2018] was used to deliver computing resources. As a remark, the proposed architecture offers a flexible solution for both independent and joint optical/wireless orchestration. The main components of the architecture included in the FUTEBOL CF are shown in Figure 3.1, and explained as follows:

- Optical and wireless controllers and agents are essential to provide programmable physical parameters, introducing a cross-layer network programmability model.

- Extended ABNO capabilities to provide controllability and observability over the new controllers lead a unified control and management architecture with a

full topology-aware.

To detail each component, in Section 3.2.3, we show how we designed and developed our Software-Defined Optical Networks inside ABNO; Section 3.2.4 shows how we designed and developed the integrated control and management with an SDR/PON solution. Finally, in Section 3.2.5, we describe how all the components were integrated into a unified control using ABNO.

3.2.3 Software-Defined Optical Networks

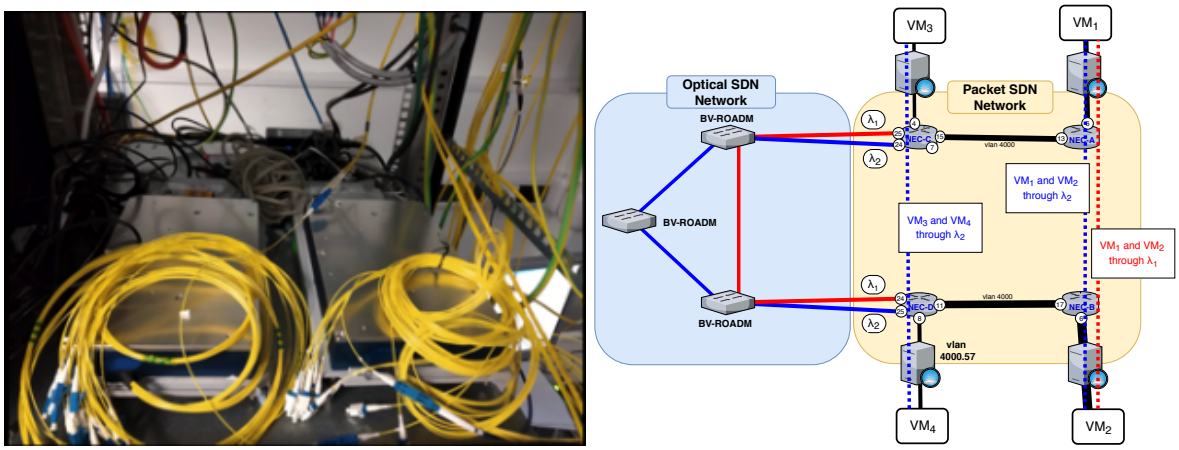


Figure 3.2: An example of using two slices separated by different λ.

To achieve the programmability of optical resources, we developed a new optical and controller abstraction for Bandwidth Variable - Wavelength Selective Switches (BV-WSS) and Optical Matrix Switch. Figure 3.2 shows how optical devices placed at the UNIVBRIS testbed, and topology with 2 slices where each slice is a combination of computing resources in a given path thought a λ_x (e.g., VM_1 to VM_2 thought λ_1). We used a total of two BV-WSS containing sixteen configurable optical ports and one common port. An Optical Matrix Switch with forty-eight was also used. Sections 3.2.3.1 and 3.2.3.2 describes our optical agent and controller implementation, respectively.

3.2.3.1 Optical Agent

As shown in Figure 3.1, we created optical agents that expose an Extreme Southbound Interface (eSBI). The optical agents are in charge of interacting with the optical devices and, further, expose the configuration by using a high-level API. The API can be consumed according to the controller's decisions. The underlying devices may have different connectivity approaches (e.g., serial port), and, therefore, there is a need to create a well-known abstraction interface to the controllers. A unified

control converts the abstraction function into the controllers and facilitates their access to different devices.

```
{
    "set":[
        {
            "id_device": 1,
            "bandwidth": 50.0,
            "spacing": 0.0
        },
        {
            "id_device": 2,
            "bandwidth": 50.0,
            "spacing": 0.0
        }
    ]
}
```

Code 3.1: Creating Grid

The Optical Agent was developed in Python 3 and is in charge of managing optical resources: Optical Switches and BV-WSS. To interact between Optical Controller, the Optical Agent uses the REST API as eSBI API. To set a λ in the optical devices by using the eSBI, we have to use the following messages: step 1, we define the grid by informing bandwidth and spacing in a given device ("id_device"). Each optical device has a unique identification. For instance, a JSON message code displayed in 3.1 shows the grid setup defined in the devices 1 and 2 ("id_serial" equal 1 and 2). In the devices 1 and 2, we set the bandwidth (i.e., channel width) equal 50.0 GHz and width spacing equal 0.

```
{
    "set": [
        {
            "id_device": 1,
            "channels": [
                {
                    "frequency": [193.85,193.95],
                    "port": 7,
                    "attenuation": 0
                },
                {
                    "frequency": [193.75,193.85],
                    "port": 1,
                    "attenuation": 0
                }
            ]
        },
```

```
        {
            "id_device": 2,
            "channels": [
                {
                    "frequency": [193.85,193.95],
                    "port": 3,
                    "attenuation": 0
                },
                {
                    "frequency": [193.75,193.85],
                    "port": 4,
                    "attenuation": 0
                }
            ]
      }
    ]
}
```

Code 3.2: Setting wavelength and attenuation

Step 2, we set to a given optical port the interval of frequency or central frequency according to the bandwidth setting, and, additionally, the attenuation control. As an example, in Listing 3.2, we are setting the parameters, as mentioned earlier, for each port in each device, wherein the device 1, port 7, we set the frequency interval from 193.85 to 193.95 and attenuation 0.0. The Optical Agent can return all the channels available after setting the bandwidth. For futher details, the optical agent code has been published at http://github.com/nerds-ufes/futebol-optical-agent/.

3.2.3.2 Optical Controller

The optical controller was developed using the Framework Ryu [Ryu 2015] version 4.23 that was extended to expose the available resources of a given optical network. The OpenFlow interface was used as a standard communication with the orchestrator ABNO. In this way, the orchestrator will only be allowed to see the resources that the experiments provisioned, avoiding the conflicts of the raw resources available for each slice. Therefore, ABNO interacts with the optical controller by using its Northbound Interface (NBI), as shown by Figure 3.1. Once the experimenters have a list of devices and ports available to experiment, ABNO installs a flow rule that defines the combined configuration of port, VLAN tag and λ_x. As an example, Listing 3.3 shows the JSON message code that attaches the port 3 and VLAN 65 to the λ_1. By using the datapath identification, ABNO can perform the configuration of each optical device exposed to a slice. To summarize, the optical controller has the following capabilities:

- returns all the optical resources available for each slice, mapped as ports in the dataplane (e.g., port 1 is mapped to a given λ_1).
- receives the flow rules and converts it in the appropriate message that is sent to the optical agents by using their eSBI.

```
{
    "dpid": 1,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "flags": 1,
    "match":{
      "in_port": 3,
      "dl_vlan": 65
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
}
```

Code 3.3: Setting wavelength and attenuation for a slice by using the optical controller

3.2.4 Software-Defined Radio

3.2.4.1 SDR/PON agent

In Figure 3.1, the SDR/PON agent is implemented through the open-source srsLTE library [Systems 2017] as the BBU part of the wireless system. Therefore, SDR/PON agent interacts with this modified system that allows dynamic reconfiguration of the bandwidth of both the Physical Downstream Shared Channel (PDSCH) and the Physical Upstream Channel. Indeed, it is achieved by reconfiguring the number of Physical Resource Blocks (PRBs) used by the signal, which affects the bandwidth, sampling rate, FFT size, and other signal processing blocks. The RRH part is implemented by using the USRP X310 reconfigurable radio device, which directly connects to the ONU through a 10G Ethernet interface. In the downstream direction, the BBU sends I/Q samples over the PON towards the USRP board, which operates digital-to-analog conversion and upconverts the signal to the 2.5 GHz ISM band. The LTE user equipment (UE) is implemented through a USRP B210 radio device, linked to a server implementing the full-stack LTE UE (also open-source from srsLTE) [Slyne et al. 2019].

3.2.4.2 SDR/PON Controller

[Slyne et al. 2019] access the SDR/PON controller located at Trinity College Dublin, Ireland, to configure the fronthaul rate of each cell. Hence, it enables dynamic physical resources reconfiguration of their wireless solution. Furthermore, the SDR/PON controller can coordinate spectrum reuse across multiple adjacent cells. The cell bandwidth can be dynamically modified as users move across the cells, where, for instance, the higher bandwidth sets to cells with higher demand. It enables efficient frequency reuse across adjacent cells, especially at the cell edges. Besides, this is the main advantage of this proposal, since the fronthaul rate of each cell over the PON is proportional to the cell bandwidth, by increasing/reducing a cell bandwidth, we also increase/reduce the cell fronthaul rate. This synergy across the optical and wireless bandwidth allocation significantly improves the statistical multiplexing properties of the entire optical-wireless system.

3.2.5 Orchestration enabled by cross-layer programmability

In order to tackle the orchestration of experiments across wireless, packet-switched, and optical network domains, ABNO architecture has been developed as a modular component in FUTEBOL CF. We extended the ABNO capabilities to interact with different northbound interfaces as access to the southbound interfaces. It has been implemented to support the orchestration of wireless, optical, and packet-switched networks. As a result, a REST API was defined as a standard interface taking into account programmable parameters for optical, wired, and wireless physical layers. This API considers the specific physical layer parameters that are exposed on top of SDN and SDR controllers. Figure 3.1 shows how ABNO gives expressiveness from different networks, once commands can be sent by using a unified REST API interface to the PON/LTE controller, Optical controller, Openflow controller, and WiFi controller. For futher details, the ABNO code has been published at http://github.com/nerds-ufes/futebol-abno-orchestrator/.

3.3 Use-case 1: an inter-testbed orchestration and control in optical and wireless networks

In this section, we present an experiment of optical and wireless spectrum allocation in SDN-controlled wireless-optical-cloud architecture (Section 3.3.2). Section 3.3.1 gives an overview and description about the testbed. Figure 3.3 highlights all the components and partners involved in the use-case. To clarify, Figure 3.3 highlights in yellow the components and testbeds involved in this use-case.

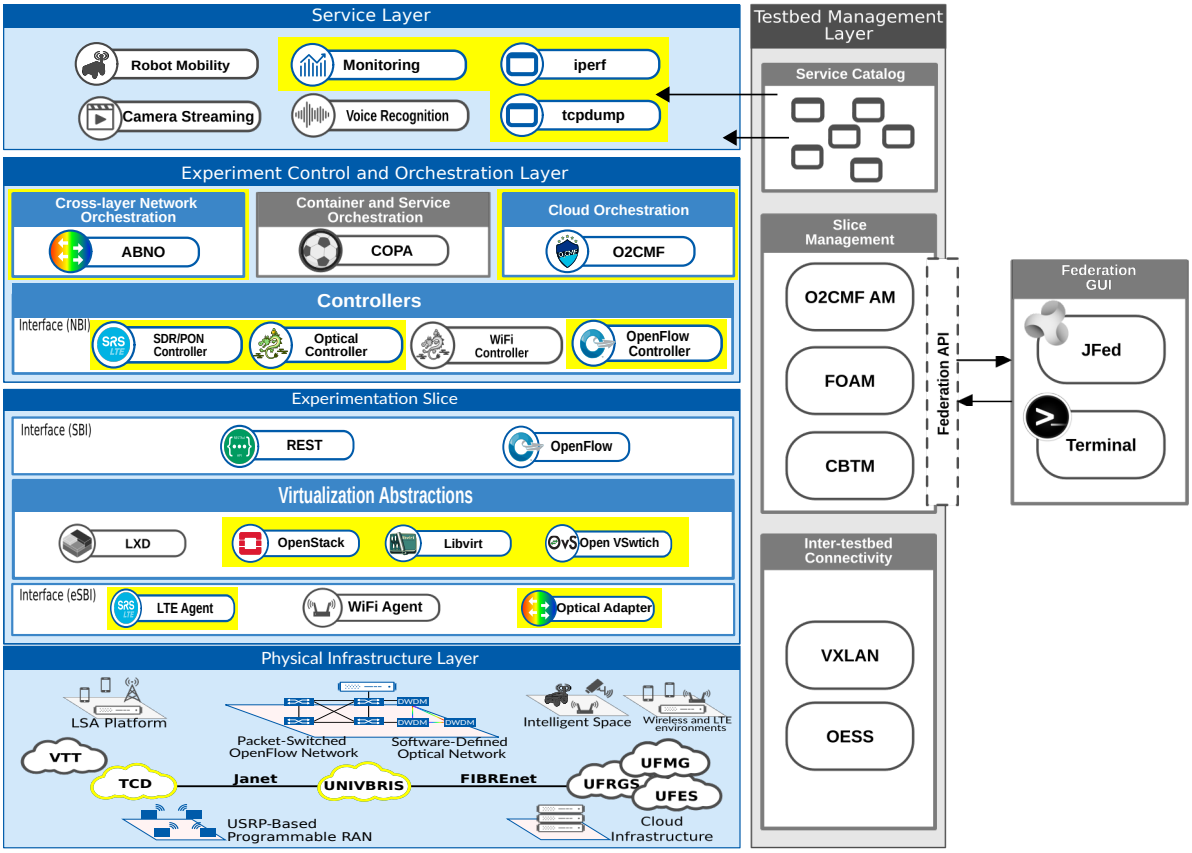


Figure 3.3: Highlighting the components used in the use-case 01: an inter-testbed orchestration and control in optical and wireless networks.

3.3.1 Testbed description

As an example of our proposed solution, by combining ABNO and SDR/PON controller, Figure 3.4 reproduces a scenario where we have two mobile users, UE_1 and UE_2 [Slyne et al. 2019]. Each UE is assigned with necessary bandwidth allocation, which is sufficient to conduct low bit rate communications, for instance, sending an e-mail or browsing. In this figure, event 0 relates to the identification and authentication of the UE devices by the EPC (Evolved Packet Core) MME (Mobile Management Entity) and HSS (Home Subscriber Server). When user A starts using a higher bandwidth service (Event 1), the increase of capacity is detected by ABNO in which it increases the mobile capacity by sending instruction through the restful API to Metro-Access SDN Controller (implemented in RYU [Ryu 2015]). It instructs the BBU to change the PRBs (Event 3) and re-configures the OpenFlow switch to guarantee capacity to the front-haul link over the PON (Event 4). When the BBU updates its wireless bandwidth, it resets the physical channel with the UE, which synchronizes to the new sampling rate, FFT size, etc. (Event 5).

We use Table 3.1, empirically derived in a previous experiment [Alvarez et al. 2018], and revalidated it in our experiment, to correlate the relationship between channel

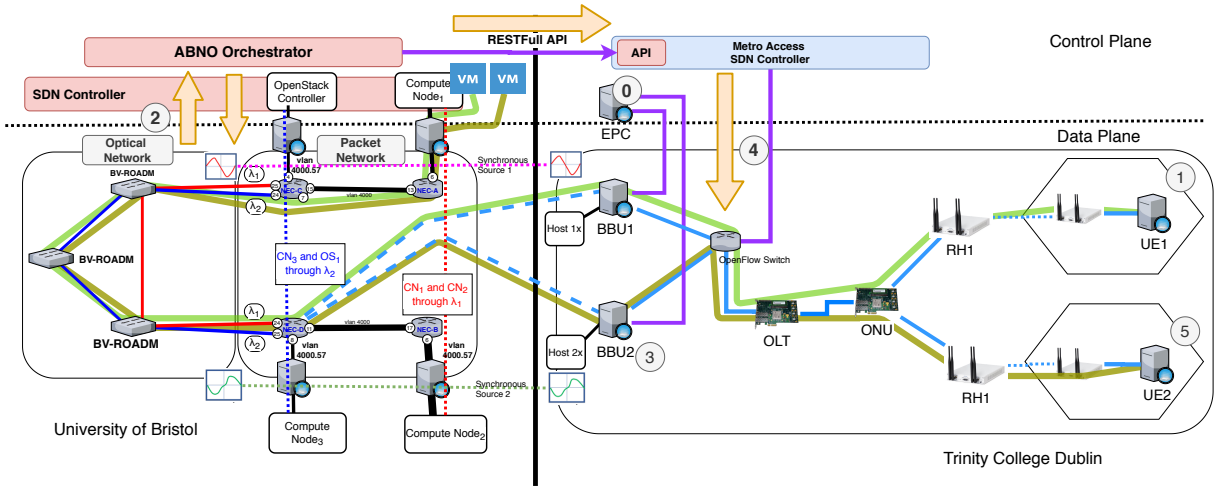


Figure 3.4: Orchestration for heterogeneous resources [Slyne et al. 2019].

Table 3.1: Bandwidth to I/Q and application rate mapping [Slyne et al. 2019].

Bandwidth	Fronthaul Rate	Max Cell Capacity
3 MHz	121 Mbps	1.97 Mbps
5 MHz	184 Mbps	15.4 Mbps
10 MHz	368 Mbps	19.5 Mbps
15 MHz	488 Mbps	21.8 Mbps

bandwidth, front-haul rate, and cell capacity. Once the data stream terminates, the Core domain controller detects a cell load that is below the current cell capacity, and it triggers the reduction of the mobile bandwidth for that cell. The additional capacity available over the PON can thus be reassigned to the other cells, together with the freed spectrum resources.

3.3.2 Proof-of-concept

For the demonstration, the optical wireless equipment is located at Trinity College Dublin, Ireland, while the ABNO, Optical controller and Optical core network are located at the UNIVBRIS, UK. The two systems will be connected live at the control plane level. Our demonstration shows the practical feasibility of synchronising dynamic resource allocation across a mobile LTE and a fixed PON system, as well as its benefit for improving the Quality of Service (QoS) of the UEs when high bandwidth-demand applications and services are provided.

During the demonstration, we will show how the controller firstly receives a report of low load and sets the number of PRBs to 15 (corresponding to a 3 MHz wireless bandwidth). Then, after the start of the video application (event 1 in Figure 3.5), the controller sets the number of PRBs to the higher rate 25 (5 MHz)

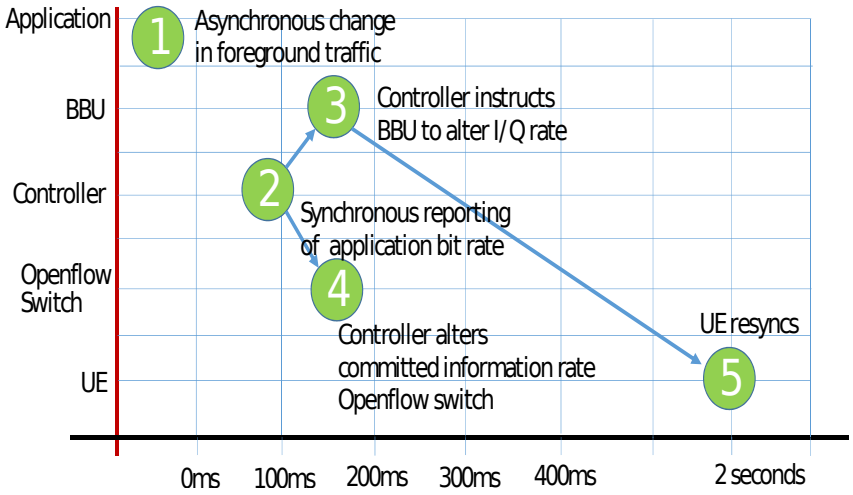


Figure 3.5: Timing Results [Slyne et al. 2019].

and reconfigures the Committed Information Rate (CIR) of the switch to cope with the extra load.

We will also provide insight into the E2E reconfiguration time of our implementation. Typically, the BBU changes the I/Q rate in 1 second and the UE resets and synchronises within 10 seconds. At this point, the LTE capacity is increased towards the UE and is made available at the application level. Indicatively, most of the reconfiguration time is spent in the reset of the data plane, due to the re-set of the PRB numbers of eNodeBs.

3.4 Use-case 2: optical, wireless and cloud slice scaling

In this section, we present an experiment where an experimenter simultaneously implements optical-wireless slice scaling, from the network infrastructure's perspective (Subsection 3.4.2.1) and vertical service scaling in the cloud (Subsection 3.4.2.2), using our components inside the FUTEBOL CF. Section 3.4.1 gives an overview and description about the testbed. Figure 3.6 highlights in yellow the components and testbeds used in this use-case.

3.4.1 Testbed description

As described by [Both et al. 2019], the Figure 3.7 portrays the physical and virtual resources allocated from three different testbeds: i) in Brazil, the Federal University

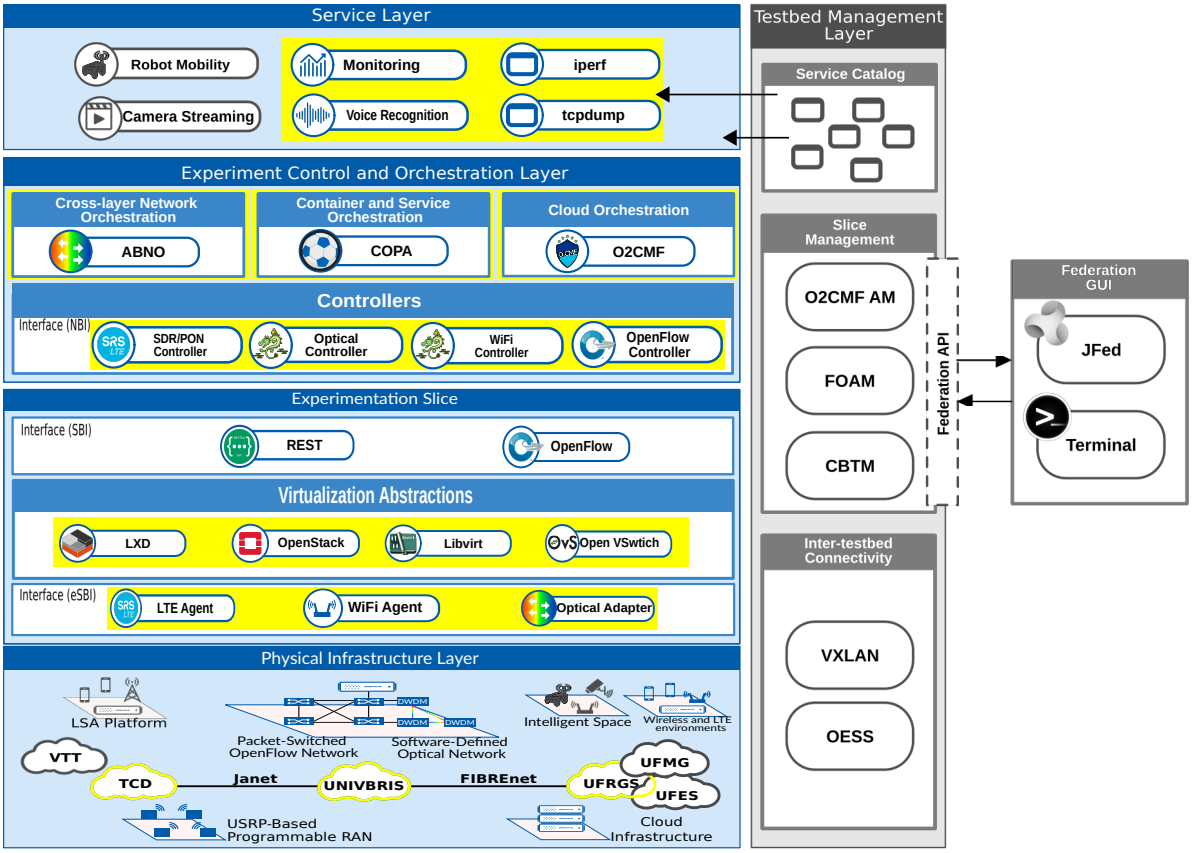


Figure 3.6: Highlighting the components used in the use-case 02: optical, wireless and cloud slice scaling.

of Rio Grande do Sul provides cloud resources using containers; ii) in Ireland, the Trinity College Dublin has developed a coordinated allocation of fiber and wireless spectrum; iii) and in the United Kingdom, the UNIVBRIS is providing the scalable meshed optical and OpenFlow networks that give the physical connectivity for an OpenStack cloud environment. ABNO is in charge to orchestrate optical and packet resources at the UNIVBRIS and wireless at Trinity College Dublin (TCD). Thus, we have extended ABNO to provide multi-resource and multi-domain orchestration capabilities.

The Federal University of Rio Grande do Sul used COPA to provides VM as Access Points managed by a WiFi-enabled SDN controller [Moura et al. 2015]. From Trinity College Dublin was provided a PON/LTE-based C-RAN, by using VM and USRPs with a full-stack LTE implementation of BBU, RRH, and UE, as well as virtualized components of an EPC (*e.g,.* vMME, vS-/P-GW, vHSS) through OpenAirInterface [Nikaein et al. 2014]. The control was carried out by a custom SDR controller and as a controller piece onto the ABNO Orchestrator. These LTE devices are interconnected to a PON fronthaul, composed of Optical Line Termination (OLT) and Optical Network Terminal (ONT). Although we could collect results from the

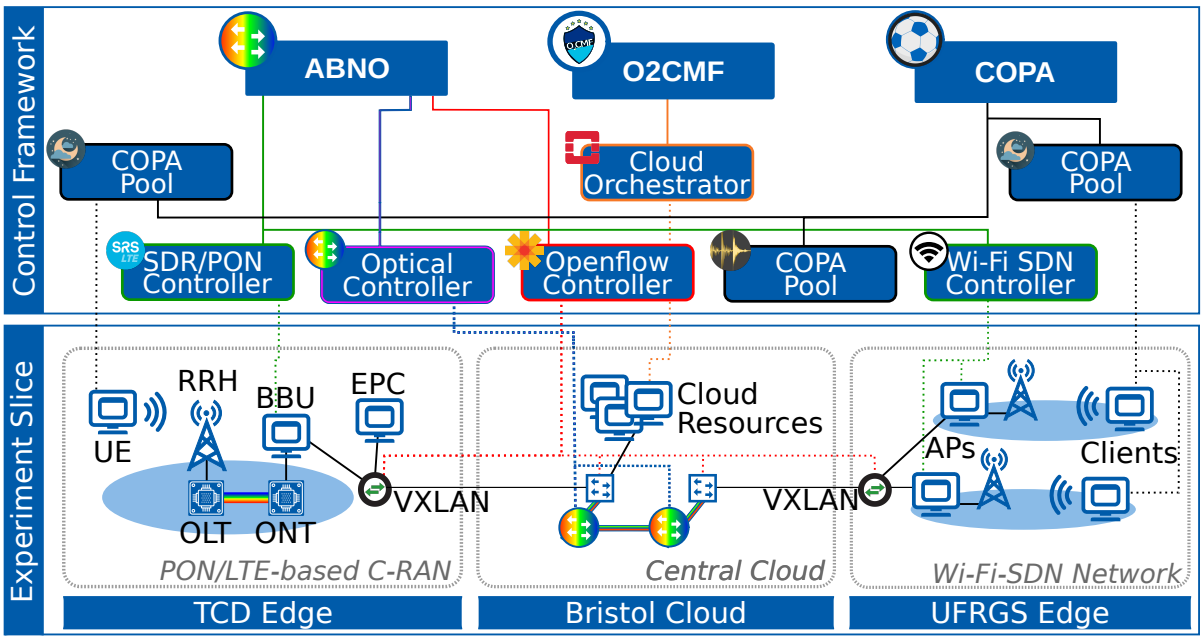


Figure 3.7: Detailed view of the control plane connectivity related to the experiment components. Adapted from [Both et al. 2019].

Federal University of Rio Grande do Sul, we decided to focus the gathering of the results from the UNIVBRIS as our central cloud and Trinity College Dublin as our edge.

Figure 3.8 shows the workflow of interactions between the Control Framework components (upper part of Figure. 3.7) to orchestrate the provisioned resources (lower part of Figure. 3.7) during the execution of the experiment. Initially, COPA is responsible for deploying a distributed service as containers in COPA Pools at cloud and edges, which is achieved in about ∼5 seconds. The COPA Pool is a component of COPA that is added to the experiment slice and allows container deployment, monitoring, and migration across testbeds. For this experiment, we employ a service that analyzes sound samples provided by mobile users worried about their quality of sleep. Users of this service utilize their smartphones (*i.e.* UEs) to record the ambient audio for a night of sleep and send these audio samples to be analyzed by the cloud service, which is done by a machine learning algorithm that identifies patterns consistent with sleep disorders [Bublitz et al. 2017].

3.4.2 Proof-of-concept and evaluation

Figure 3.7 depicts the physical and virtual resources allocated from three different testbeds in Brazil (UFRGS), Ireland (TCD), and UK (Bristol). Inter-testbed connectivity is achieved through an overlay network (VLAN tagged). Over this network, we established VXLAN tunnels to encapsulate our experiment's control and data traffic. In this experiment, UFRGS and TCD represent edge computing, while Bristol

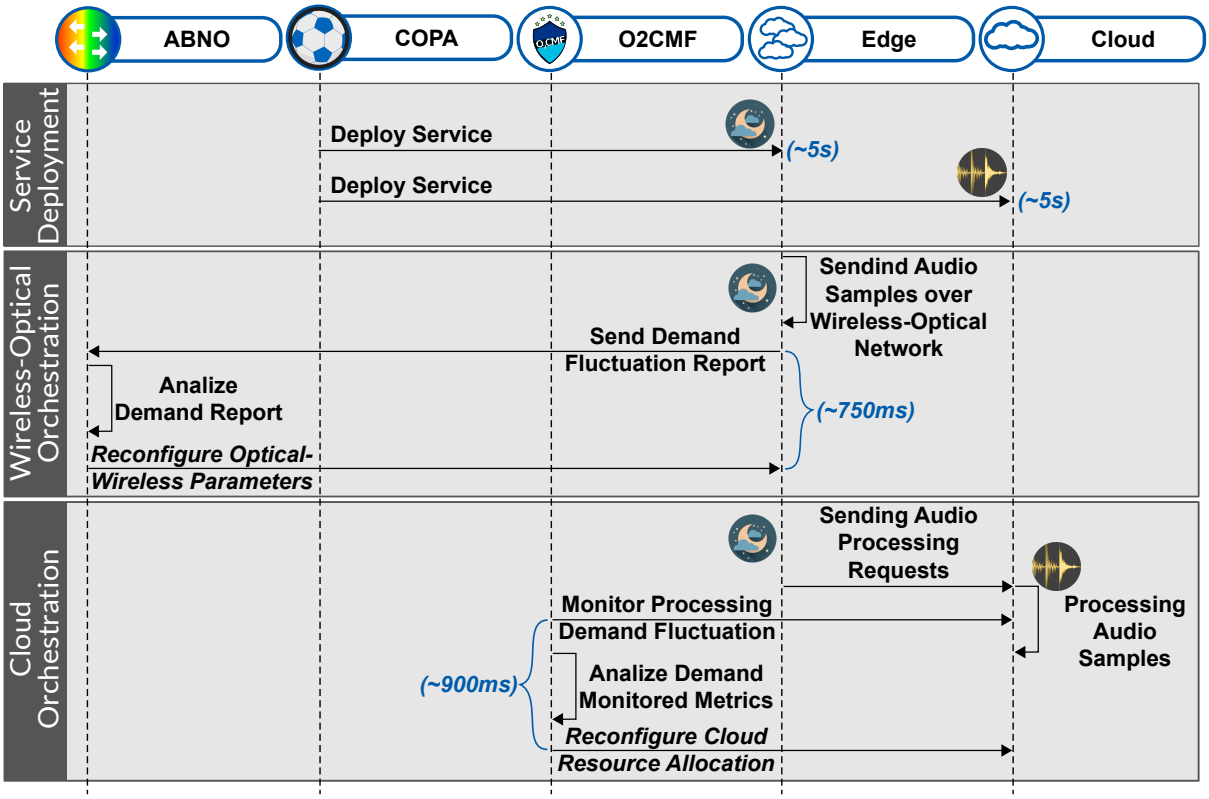


Figure 3.8: Experiment components interaction workflow [Both et al. 2019]

characterizes a central cloud. As a rough estimate, the time required to provision this experiment slice and the associated Control Framework components is approximately five to ten minutes.

The cloud computing resources in the UNIVBRIS testbed can be provisioned on demand optical/packet SDN networks. UNIVBRIS also provides cloud resources by using O2CMF over an OpenStack installation. The most processing-intensive parts of the experiment are run from this testbed. Moreover, we can dynamically assign a flow that ingresses the packet network towards the computing resources of the central cloud, or forward traffic between the two edge/fog networks via an optical switched network. To control these resources, we deployed two SDN controllers, one for packet networks based on standard OpenFlow, and another for optical networks based on an extended version of this same standard, which allows controlling physical parameters of the optical network (*e.g.*, central frequency and bandwidth of each lightpath). UFRGS provides VM as Access Points managed by a Wi-Fi-enabled SDN controller [Moura et al. 2015]. TCD provides a PON/LTE-based C-RAN, by using VM and USRPs with a full stack LTE implementation of BBU, RRH, and UE, as well as a software implementation of the EPC (*e.g.*, MME, S-/P-GW, HSS) from OpenAirInterface [Nikaein et al. 2014] managed by a custom SDR controller. These LTE devices are interconnected to a PON fronthaul, composed of Optical Line Termination (OLT) and Optical Network Terminal (ONT). Since UFRGS and TCD play

the same role of edge/fog computing for the experiment, and TCD implements a more realistic and interesting optical-wireless scenario, for the sake of brevity, the results reported in this section focus on the Bristol central cloud and TCD edge.

Figure 3.8 shows the workflow of interactions between the Control Framework components (upper part of Figure 3.7) to orchestrate the provisioned resources (lower part of Figure 3.7) during the execution of the experiment. Initially, COPA is responsible for deploying a distributed service as containers in COPA Pools at cloud and edges, which is achieved in ∼5 seconds. COPA Pool is a component of COPA that is added to the experiment slice and allows container deployment, monitoring, and migration across testbeds. For this experiment, we utilize a service that analyzes sound samples provided by mobile users worried about their quality of sleep. Users of this service utilize their smartphones (*i.e.* UEs) to record the ambient audio for a night of sleep and send these audio samples to be analyzed by the cloud service, which is done by a machine learning algorithm that identifies patterns consistent with sleep disorders [Bublitz et al. 2017].

In this experiment, we emulate demand from clients activating the service over 24 minutes. Demand initially increases from 1 to 50 clients, and then decreases back to 1, creating a demand fluctuation. As audio samples are sent over the wireless network, traffic demand fluctuates at the edge, requiring ABNO to take action and adjust optical-wireless resource allocation. As requests reach the service hosted in the cloud, processing demand also changes, activating the cloud vertical scaling feature provided by the O2CMF orchestrator to adjust the available resources automatically.

3.4.2.1 Scaling optical-wireless resources

We evaluate the slicing of physical and virtual resources located in the TCD edge, and orchestrated according to the service demand in Bristol's central cloud. A typical scenario might be that of a mobile user on an LTE network requesting dedicated bandwidth through a core and fronthaul network for a pre-determined time. We have previously demonstrated slicing and scaling of capacity in a converged fixed-mobile network through variable rate LTE over a PON fronthaul [Alvarez et al. 2018].

We designed a customized SDR/PON controller for the converged optical and wireless domains that assess the cell capacity required to provide a particular level of fronthaul application bandwidth, and coordinate in real-time the allocation of this capacity among the BBU, RRH, and PON, achieving ∼750ms reconfiguration time. This SDR/PON controller provides a RESTful API to connect to ABNO, which specifies optical parameters for provisioning the PON slice, as early mentioned in Section 3.2. Moreover, ABNO requests changes in the wireless configuration to the

SDR controller, which reconfigures the bandwidth in the cell. These changes result in an increase or decrease in the LTE network capacity, which ABNO sends to the SDR controller to change the bandwidth used by the cell using LTE-specified Master Information Block messages, reconfiguring its sampling rate. LTE cell bandwidth is usually fixed by mobile operators, typically at 20 MHz, and requires a fixed transmission of I/Q samples up to a rate of 730 Mbps over a dedicated optical fiber. In our scheme, only the used cell bandwidth needs to be provided, so that the freed-up PON capacity can be re-used by other adjacent LTE cells, or by other lower-priority traffic.

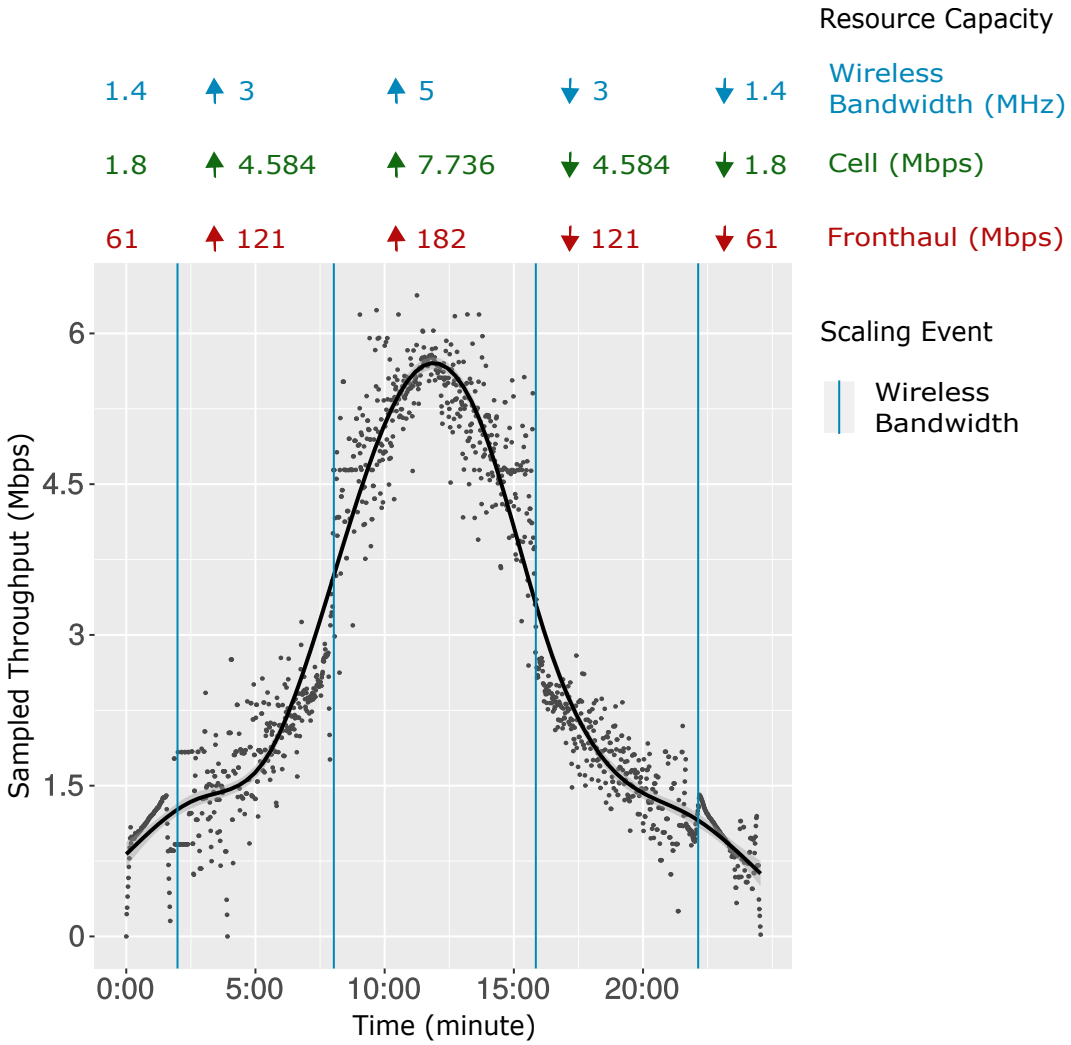


Figure 3.9: Orchestration of the physical optical-wireless resources [Both et al. 2019]

Figure 3.9 shows the configuration of the wireless bandwidth and measured dynamics of the maximum rate for cell and fronthaul. The LTE network scaling is shown in response to the fluctuation in the mobile users demand for the service hosted in the cloud. Initially, the bandwidth is set to 1.4 MHz, corresponding to the minimal cell bandwidth and minimal fronthaul capacity, since demand is at its

minimal ($\sim$ 1.7 Mbps). As more clients activate the service, ABNO dynamically scales the backhaul bandwidth to 3MHz and 5MHz, achieving a maximum throughput of over 6 Mbps. As demand decreases, these changes revert to the initial state. The fluctuation in service demand not only triggers the reconfiguration of the optical-wireless capacity in the edge, but also impacts the demand for the application running in the cloud, as explained next.

3.4.2.2 Vertical scaling in the cloud

In our experiment setup, the service is provided by a Web server virtualized in a container deployed by using COPA. The container runs in a server hosted by a cloud infrastructure provided by O2CMF. In our experiment setup, both the service and the mobile application are deployed in containers using COPA Pools. The COPA Pool is a component of the COPA architecture that is added to the experiment slice and allows deployment and migration of containers across testbeds. As in the real-world counterpart, clients can use the service and have their audio samples processed over the time they are connected. Hence, O2CMF automatically scales the COPA Pool's resources in response to perceived demand; orchestrated resources are RAM, from 1 to 4 GB, and virtual CPUs, from 1 to 4. Results for the experiment are presented in Figure 3.10, where the time evolution is registered in the lower x-axis, and the clients' demand in the upper x-axis; the y-axis shows the processing time for requests performed. Scaling events, *i.e.*, time instants when O2CMF orchestration increases or decreases the server's resources, are marked with vertical lines, and the status on the allocation of resources is depicted in the upper part of the graph. Overall, O2CMF orchestration takes about 7 seconds to complete the whole vertical scaling process as shown in Figure 3.8, including the thresholds monitoring. The vertical scaling itself takes about 900 ms.

As shown in Figure 3.10, the mean processing time remains roughly constant from the start of the experiment, with just one client connected, to the eight-minute mark, when 30 clients are served. The following increase in clients from 35 to 50, observed from the eighth to the eleventh minute, results in a significant increase in the mean processing time. The result indicates that the cloud orchestration is efficient in providing CPU units when demand increases; a decline in performance (*i.e.*, higher response times) is perceived only after the stipulated maximum allocation (4 CPUs) is reached. Moreover, the experiment uses a memory-intensive service, RAM utilization rapidly increases with the constant influx of new clients, ultimately resulting in service failures marked with red X's in Figure 3.10. This behavior can be explained by the chosen parameters for orchestration, such as the minimal interval between scaling events (6 seconds) and the fixed amount of resources scaled in each

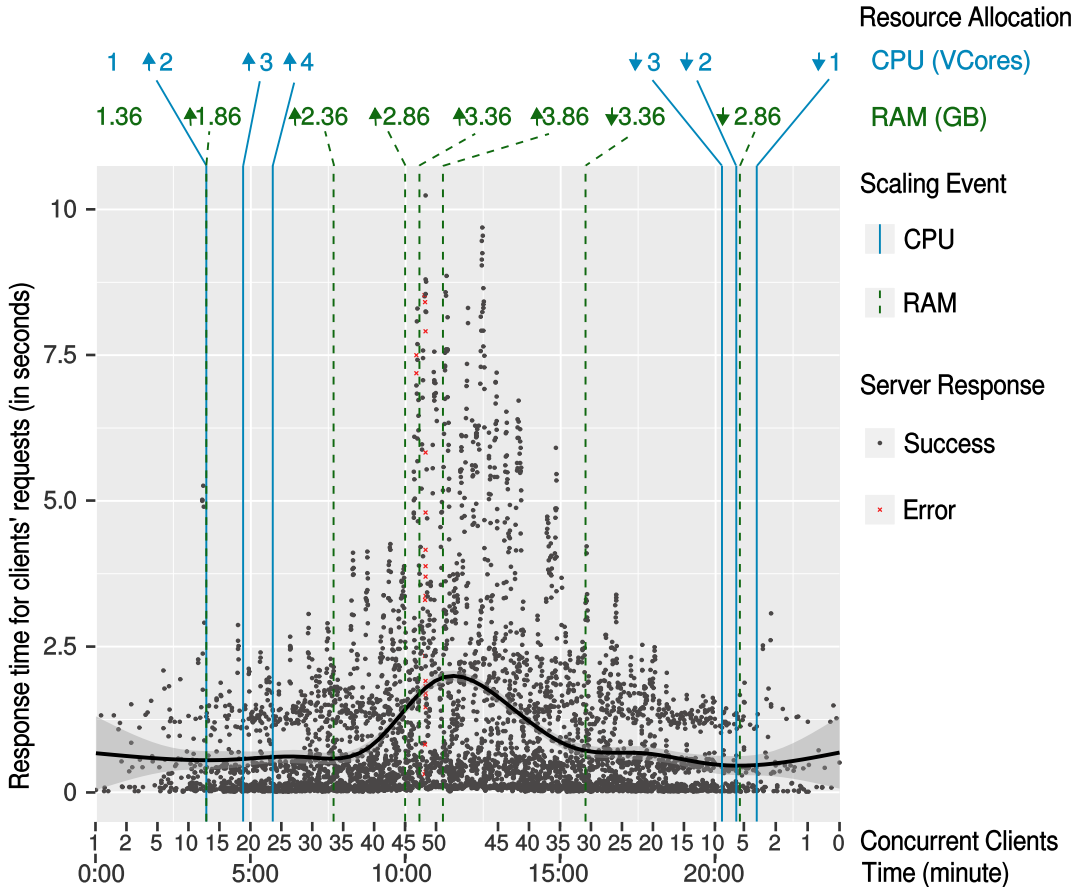


Figure 3.10: Orchestration for automatic service for the Cloud resources [Both et al. 2019]

event (0.5 GB for RAM; 1 virtual core for CPU). Chosen orchestration parameters thus indicate a Cloud tendency to lean towards resource-saving, at the expense of service availability.

As the experiment uses a memory-intensive service, RAM utilization rapidly increases with the constant influx of new clients, ultimately resulting in service failures (red X's in Figure 3.10). RAM insufficiency results in service inability to process incoming requests, which occurs when the scaling is delayed, as seen by the ten-minute mark. Note that all failures occur when demand increases to 45 clients, and not afterwards when it reaches 50. Although the reconfiguration time is fairly small, taking ∼900 ms (Figure 3.8), this behavior can be explained by the parameters chosen for the orchestration algorithm developed for this experiment, such as the minimal interval between scaling events (6 seconds), the usage threshold for scaling resources up (80% for both CPU and RAM), and the fixed amount of resources scaled by each event (0.5 GB RAM; 1 virtual CPU). These parameters indicate an orchestrator tendency to lean towards resource-saving, at the expense of service availability. The experimenter can work the trade-off between resource-saving and service availability by adjusting the parameters with O2CMF.

The experiment investigates the efficient allocation of cloud resources, assisted by the metrics provided by COPA. Throughout the experiment, the average resource allocation for the server was of 2.97 virtual CPUs, and 2.71 GB of RAM. A cloud with no scaling support would require constant allocation of 4 virtual CPUs and 3.86 GB of RAM. The dynamic allocation represents an average resource-saving of 1.03 virtual CPUs and 1.15 GB of RAM over the the experiment duration. This surplus could thus be reallocated and the resulting resource-saving credited for the underused service, *e.g.*, depending on cloud's service agreement. Orchestration aggressiveness, *i.e.*, how quickly (and by how much) the cloud scales the resources, plays a significant role in this and should be fine-tuned according to each application requirements (*e.g.*, high availability applications may require aggressive scaling up, and conservative scaling down, of resources).

Hence, in this experiment, the FUTEBOL CF enabled the network scaling orchestration between the optical-wireless resources to fit the spectrum utilization with the traffic fluctuations from clients activating/deactivating a sample service, and the corresponding vertical scaling of resources in the cloud to cope with the increase/decrease of incoming audio processing requests.

3.5 Chapter Remarks

The proposed architecture was integrated in the The FUTEBOL CF, which facilitates experimentation across optical, wireless, and cloud domains. ABNO plays an essential role of cross-layer orchestrator by interacting with the network controllers in optical and wireless resources, whereas COPA and O2CMF are orchestrator tools for computing resources, which, for instance, can enable the monitoring and live migration of light VNFs across heterogeneous testbeds. We have designed and developed the extension tool for orchestration in the FUTEBOL CF, adding new functionalities to enrich to the experimenters an integrated research over optical and wireless networks. Future discussions involve solution for new critical applications that bring specific requirements, such as zero mobility interruption time. Hence, it can provide a flexible and proactively solution aiming, for example, ultra-reliable and low-latency handover, to perform the reconfiguration as fast as needed to meet the E2E requirements [Martínez et al. 2018, do Carmo et al. 2019b, Guimaraes S et al. 2020].

Chapter 4

Orchestration for seamless, reliable and low-latency mobility

4.1 Overview

The next-generation networks will need to deal with a new set of groundbreaking applications such as intelligent transportation systems, tactile Internet, and cyber-physical systems (CPS) for Industry 4.0. Among the major of service groups, the ultra-reliable and low-latency communications (URLLC) aim to foster new applications stringent demanding on latency and reliability [Pocovi et al. 2018]. Nevertheless, notwithstanding their promising capabilities, those networks also expect to coexist with different resources and technologies, such as WiFi and LTE networks crossing optical layers.

Despite all innovative features brought by this new breed of wireless systems, it is clear that URLLC and other services will coexist, complement, or even face competition from different technologies, such as WiFi and LTE. The scientific community noticed this trend early and is already pushing WiFi in such direction with a series of new standards to meet requirements similar to 5G systems [Ayyash et al. 2016].

Off-the-shelf WiFi equipment is already used to consistently offload a considerable part of the 4G and 5G mobile traffic. Besides, WiFi has potentially better performance in indoor environments than broadband wireless networks for low-speed mobility cases. Little has been done, however, to equip current low-cost off-the-shelf WiFi with extra functionalities to bring it closer to 5G new requirements. Similarly, WiFi could be exploited in the future to fulfill the requirements of many – less demanding – emerging applications, offloading 5G networks and, as a result, significantly reducing cost [Bayhan et al. 2018].

There is, however, beg the question regarding WiFi co-existence/competition for serving URLLC indoor applications: the non-existing native WiFi orchestration

mechanisms with heterogeneous network and systems behind them. Mobile systems have all-encompassing solutions bridging across radio heads and wired layers of the network. This integration is essential, such as for handover tasks; and, more recently, to provide efficient access to the cloud. The latter is an essential piece of the new communication puzzle since clouds will not only host most of the user's applications and data, but also critical network services will be virtualized there.

The use of software-defined networking (SDN) and network function virtualization (NFV) are of great help in achieving control integration and also the management of several physical resources. The goal is meeting E2E requirements of these services that foster the innovative applications expected by next-generation networks [Both et al. 2019]. For instance, macro-diversity techniques are used in mobile networks to deal with the slow fading phenomena in wireless channels (e.g., path loss and shadowing) and with cell failures [Li et al. 2018]. These techniques enable multiple links to different radio access points, which may use the same access technology or combine different technologies. By using data duplication and multiple transmission from the base stations, the macro-diversity helps to increase the reliability of the wireless communications. It provides robustness to the systems during handover and fault recovery processes [Nielsen et al. 2018].

Although macro-diversity and multi-connectivity schemes are widespread in current mobile networks, present, and future IEEE 802.11 standards, do not leverage such techniques appropriately. Thus, it seems that WiFi is unprepared to meet requirements from upcoming real-time applications such as Industry 4.0 or eHealth verticals [Mello et al. 2019]. Nevertheless, there is still a vast and mostly untapped potential in orchestrating SDN and NFV alongside conventional WiFi. These wireless network solutions do not present an efficient handover scheme and also suffer from interference and noisy channels. It hampers performance and imposes prohibitive packet loss rates for real-time applications. This poor performance, in terms of reliability and latency, can be improved by implementing multi-connectivity schemes.

This chapter argues that the combined orchestrability and control actions may unleash comprehensive, reliable, and low latency mobility management functions in low-cost off-the-shelf WiFi devices. We also experimentally demonstrate an orchestration and control of multi-connectivity WiFi solutions for seamless handover, using a new context network approach (e.g., SNR and computer vision) to achieve a better high traffic density industrial environments [do Carmo et al. 2019b, Guimaraes S et al. 2020]. The solution is demonstrated by enabling a challenging cloud-robotics application with real-time remote control and mobility requirements.

4.2 Proposal

This section describes the potential limitations, the design principles, and the architecture of our following proposals seamless and reliable handover orchestration in off-the-shelf WiFi.

4.2.1 Design Principles

The design principle of our proposal SDN/NFV architecture was motivated by the following challenges in enabling the management of WiFi networks to balance reliability and traffic density trade-offs during handovers while optimizing the throughput capacity of each cell.

To address the early-mentioned challenges, we design the solution based on the following integrated enablers:

- a source-routing solution for E2E communication including cloud, wired, and wireless resources;
- packet duplication for reliable wireless communication; and
- seamless handover and failover schemes with zero mobility interruption time (MIT).

All the integrated enablers are to form a coordinated orchestration that performs the network reconfiguration that meets the requirements of an E2E communication.

4.2.2 Architecture Design

The design principles of our SDN architecture are motivated by URLLC requirements for new real-time applications in off-the-shelf WiFi environments. The core of our architecture is based on the splitting of WiFi functions to reverse the roles of traditional WiFi equipment and provide AP functionality to a mobile element [Martínez et al. 2018]. The path diversity relies on an infrastructure cross-connecting cloud, wired and wireless environments. We detail the full architecture and its operation in the following sections.

4.2.2.1 Multi-connectivity: Splitting WiFi functions

Figure 4.1 illustrates the functional components of the architecture. It is composed by three main blocks: i) the specific and individual mobile element (e.g., a robot or IoT device) and its wireless communication module; ii) the wireless access network and the backhaul network, which are responsible for routing and forwarding data

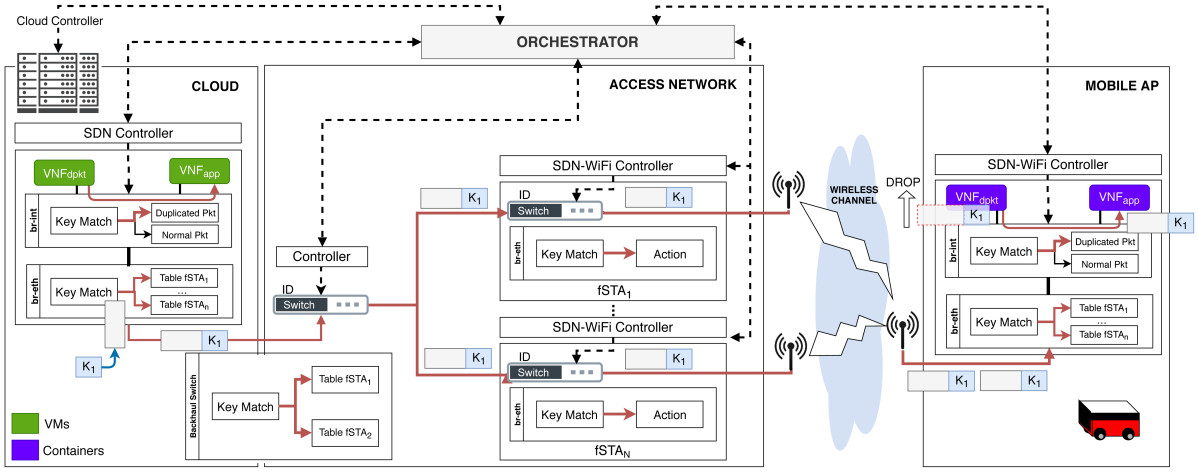


Figure 4.1: Architecture Design: handover and duplication routing.

packets during mobile-cloud communication, and; iii) the edge cloud platform, composed by a Virtualized Infrastructure Manager (VIM) environment, responsible for providing services to a specific group of mobile devices. These components are managed by an orchestrator with a global view of the system, also hosted in the cloud.

a) *Mobile Access Point (mAP)*: Through the use of SDN techniques, it is possible to provide mobility to the AP of a WiFi infrastructure. This change in role allows the mobile element to establish and to manage multiple associations with other devices. This modified communication module hosted in the mobile element is referred to *mobile Access Point* (mAP). Similarly, the wireless devices composing the infrastructure bear the role of *fixed client stations* (fSTAs), which are the wireless access network in our architecture. This special feature creates the possibility of multi-connectivity through association of the mAP module and different fSTAs. An SDN-WiFi controller placed in the mAP module provides a control communication channel between the wireless element and the architecture orchestrator. Moreover, a virtual switch connects the wireless domain and container-enabled mobile applications in a docker environment without modifying the IEEE.802.11 standard stack.

b) *Access/Backhaul Infrastructure*: The radio access network is composed by elements operating as fSTAs. The multi-connectivity scheme is established by authenticating and associating all nearby fSTAs to the mAP. Software agents installed in each fSTA continuously monitor the wireless channel. This information is sent via SDN-WiFi controllers to the architecture orchestrator. It then uses this information to manage mobility in a network-centric approach. Finally, the fSTAs share an OpenFlow-enabled backhaul network, in which routing and forwarding tasks are carried out.

c) *Edge Cloud Platform*: Both user applications and orchestration applications (i.e., control and management) are virtualized in a cloud platform. The cloud controller manages the interaction of VNFs with hardware resources and communicates with the orchestrator to manage routing.

4.2.2.2 SDN Source-Routing for Unicast/Multicast Communication

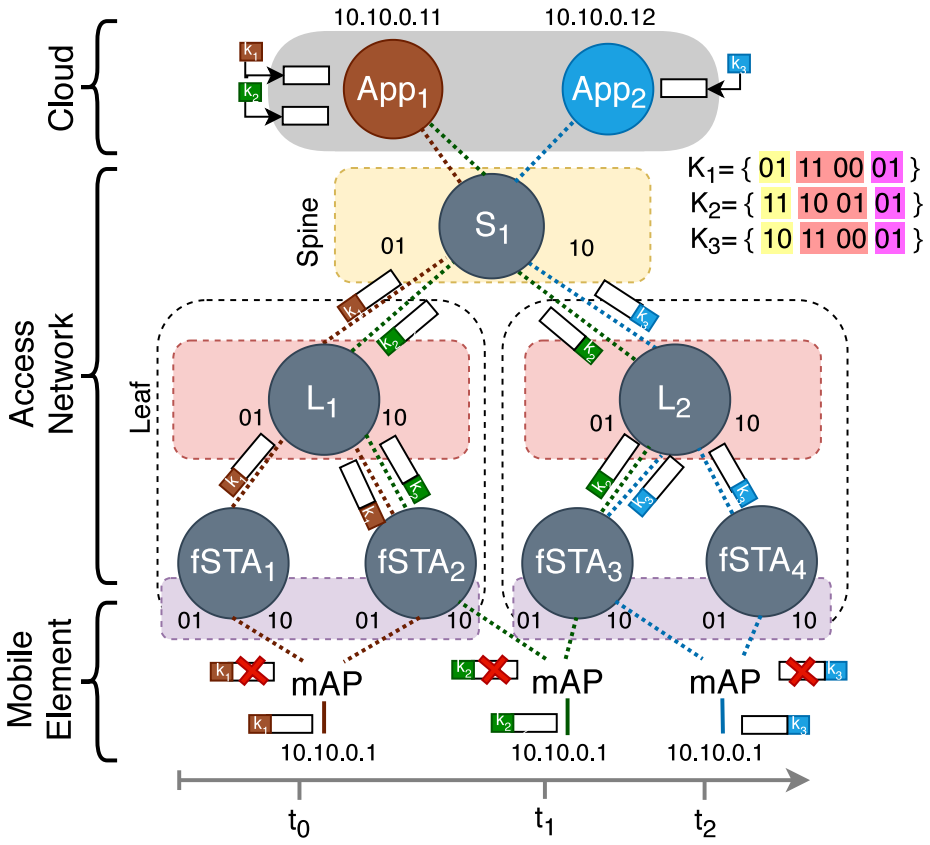


Figure 4.2: Illustrating the encoding design for unicast/multicast communication.

In order to exploit this multi-connectivity scheme more efficiently, multipath routing mechanisms enabled. For instance, Figure 4.2 illustrates an mAP that initially obtains the fSTA_1 and fSTA_2 associations at time t_0 and is then handed over to different covered areas (i.e., times t_1 and t_2) as it moves around. Given some available metric (e.g., SNR), path diversity can be leveraged to route data packets through the – current – best wireless association. Thereby, the re-routing of packets handover the connection of the mAP to a more suitable fSTA. Furthermore, communication reliability can be improved by packet duplication over this multi-connectivity scheme.

To illustrate the E2E communication setup, consider the scenario depicted in Figure 4.2 consisting of two cloud apps (App_1) and (App_2). Each app has its own E2E communication requirements. For instance, App_1 deals with the demand from the

area of the leaf L_1, whereas App_2 deals with the leaf L_2. Therefore, parallel paths (spatial diversity) can be exploited from App_1 to the mAP at the instant t_0 within its coverage area ($fSTA_1$ and $fSTA_2$ both on L_1), whereas at the instant t_1 its covered areas will be $fSTA_2$ on L_1 and $fSTA_3$ on L_2.

A codification approach has been designed to exploit the multiple paths by using source-routing schemes likewise to [Shahbaz et al. 2019, Liberato et al. 2018]. For every fSTA, there is a sequence of bits (i.e., bitmap) representing a single fSTA (e.g. $fSTA_1$ = 01). Each bitmap consists of a set of output ports according to a tag.

4.2.2.3 Multipath Routing

The exclusive-or (XOR) logical operation is used to define the tag, which represents an aggregation of output ports in each layer ($fSTA_1 \oplus fSTA_2$ = 11). The design has considered a three-tier multi-rooted CLOS topology composed of one spine (S_1) and two leaves (L_1 and L_2) as shown in Figure 4.2, limited by two nodes per layer .

For instance, a bitmap of K_2 is obtained as 11 10 01 01 in which bits b_7 to b_6 represent the result of the operation $L_1 \oplus L_2$ and, by using for it the right-most significant bit 1, b_5 to b_4 represent the result of the leaf L_1 ($fSTA_3$ = 10), and bits b_3 and b_2 represent the result of the leaf L_2 ($fSTA_3$ = 01). The bits b_1 and b_0 indicate the wireless port (i.e., 01 for 5 Ghz and 10 for 2.4 Ghz). Hence, K_2 indicates the multiple paths for the data flow that cross different spines and leave, reaching the mAP in a specific wireless frequency. Furthermore, K_1 and K_3 have the following bitmaps, 01 11 00 01 and 10 11 00 01, in which packet duplication is set by using the bits b_5 and b_4, however, forwarded to different leaves (bits b_7 to b_6). The tags are defined by Openflow rules at the virtual switches in the cloud (hosting Apps). The packets are identified by their group specific K_i tags, using the encoded value, which indicates the path for routing the data flow. The tag is implemented at the Type of Service (ToS) field in the IP packet header. To carry out the routing, the orchestrator must first determine a preferred route to reach the mAP based on the information gathered by the fSTA's monitoring agents. Then, each outgoing packet is tagged – represented in Figure 4.1 by the K_1 tag – by using an encoded value to indicate to the backhaul switch the port route of the data flow. Each port in the backhaul switch is linked to one fSTA, thus this routing schemes ultimately determines through which fSTAs the data must flow. The tag is implemented at the Type of Service (ToS) field in the IP header.

4.2.2.4 Packet duplication approach

For packet duplication schemes in WiFi networks, our architecture implements a VNF-based solution, and this fundamentally differs our proposal from previous ap-

proaches exploiting duplication of packets such as [Rentschler and Laukemann 2012, Cena et al. 2018]. Our approach leads to a hardware-agnostic solution with more agility in the creation of future services. When packets are forwarded according to the SDN Source-routing scheme explained previously, a VNF at the destination checks whether the tag represents a duplicated or a single flow. Once the data plane identifies the type of flow, it determines how the packet is forwarded: i) with the packet duplication tag, the data plane sends the traffic to be treated by the VNF, represented by the VNF_{dpkt} at the mAP in Figure 4.1; ii) with the normal packet identification, the data plane redirects the traffic strictly to the destination, VNF_{app}.

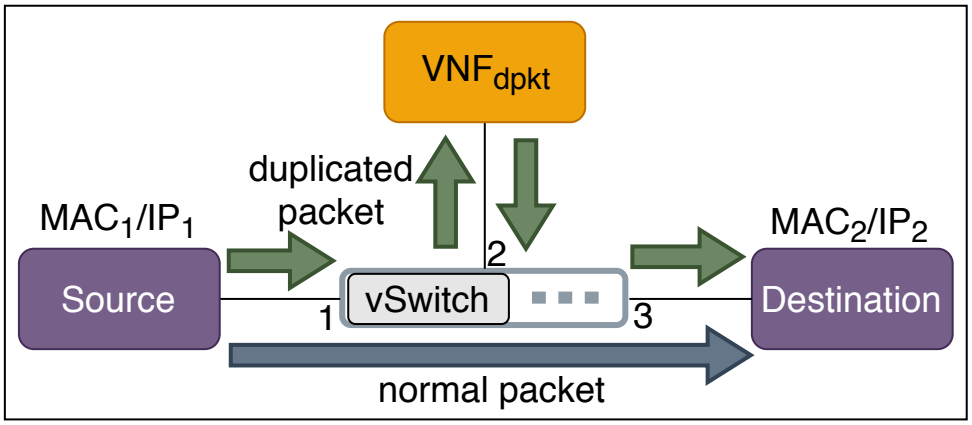


Figure 4.3: SFC operation mode.

4.2.2.5 VNF implementation

Packets are analyzed and forwarded according to their classification when in the mAP or in the cloud. Whenever a packet is either in mAP or in the cloud, it is analyzed and forwarded according to its classification as shown in Figure 4.3. Then the data plane decides if a given packet must pass through the VNF_{dpkt}. The VNF in the chaining is responsible for the flow de-duplication function performed by a group of OpenFlow rules previously defined. The VNF_{dpkt} receives a packet p_i, generates a hash $h(p_i)$ by using the TCP/IP headers, and compares if the current hashed packet $h(p_i)$ already exists in a circular list β. If so, then the packet p_i is automatically dropped, otherwise, $h(p_i) \notin \beta$, then the packet is forwarded to its destination and $h(p_i)$ added to β. Each hash inserted in the circular list β has its own time-to-live to limit the size of the list, which avoids bottlenecks in the matching process.

4.2.3 System Operation driven by Orchestrator Decisions

Tackling better decisions implies getting as much as possible well-coordinated information from different network resources in a centralized view. For this purpose,

suitable to the proposed architecture, our Orchestrator Architecture was developed being in mind to control three kinds of resources, as shown in Figure 4.1: SDN, WiFi and Cloud. The Orchestrator has a view of the entire network as SNR of each mAP, traffic density, and computing resources usage, considering the impact of each action on the network as a whole.

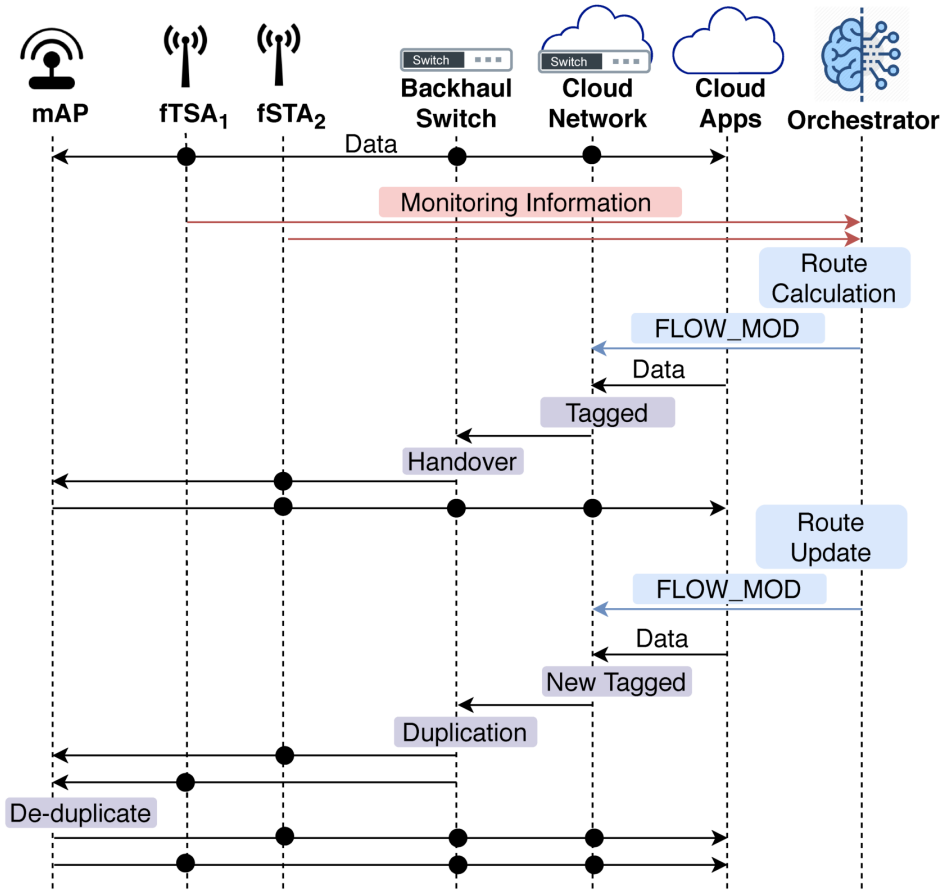


Figure 4.4: System Operation.

In the proposed architecture, an orchestrator is responsible to control SDN, WiFi and cloud resources, as illustrated in Figure 4.1. The orchestrator centralizes information on the entire network, such as SNR in wireless nodes, current traffic density, and computing resources usage, considering the impact of each action on the network as a whole. The orchestrator can be configured to use any suitable algorithm to define the best route. It opens the possibility to make accurate decisions in defining E2E routes (e.g. satisfying the requirements regarding communication from a cloud application to a mobile application in the mAP).

The system operation detailed in Figure 4.4 illustrated an initial condition where two fSTAs are associated with the mAP and the traffic is routed through the $fSTA_1$. The agents in each fSTA send the monitoring information to the orchestrator using a common Message Queuing Protocol (AMQP) interface. In this example, the orchestrator considers the SNR as the main metric for decision making. With the

SNR information relating the devices, the orchestrator calculates the best route (i.e., the fSTA with the best link conditions). Thus, the orchestrator modifies the rule in the network infrastructure to tag the packets with the encoded value of the selected fSTA. The tagged packets are analyzed on the backhaul switch to be forwarded through the new fSTA. As exemplified in Figure 4.4, the mAP then receives the data flow through the $fSTA_2$, completing the handover process.

The packet duplication process is performed when the orchestrator determines that the link conditions of the associated fSTAs do not guarantee the necessary reliability of the communication, in the presence of a lossy wireless channel. In this case, the source routing tag is encoded with the information of the fSTAs, which will perform the redundant transmission to increase the probability of successful delivery to mAP. In Figure 4.4, the traffic is forwarded by $fSTA_1$ and $fSTA_2$, and once received by the mAP, it is delivered to the VNF to perform the de-duplication function that finally forwards it to the mobile application. The answer is also duplicated in the mAP and the same de-duplication process is carried out in the cloud platform.

4.3 Testbed description

The main functional blocks that compose our architecture have been developed and tested in a real-world testbed. The mAP is installed in a Raspberry Pi 3B with the Hostapd application to enable AP functionality, equipped with one Ralink Technology RT5572 802.11 a/b/g/n wireless adapter. The access network (i.e., the fSTAs) consists of Linux PCs (Ubuntu 16.04) equipped with a wireless card based on the Qualcomm Atheros AR9300 chipset. In the backhaul, the fSTAs are connected via an Ethernet interface to an Supermicro OpenFlow-enabled switch. The architecture orchestrator manages and controls the mAP and fSTAs through SDN-WiFi OpenFlow controllers using the Ryu 4.20 Framework. The cloud platform is implemented with OpenStack Queens.

The prototyping of our solution has been developed and tested in a real testbed scenario composed of one mAP and four fSTAs. Each fSTA and mAP has an SDN and WIFI controller; both are created by using the Ryu Framework [Ryu 2015]. The mAP is a Raspberry Pi 3 device that uses hostapd to provide AP functionalities and is equipped with one Ralink Technology RT5572 802.11 a/b/g/n wireless adapter. The fSTAs are based on PC with Ubuntu 16.04 and have one Qualcomm Atheros AR9300 wireless network adapter. We performed the experiments setting up the wireless network in 5 GHz, operating in 802.11n mode, and selecting less noisy channel according to the spectrum utilization.

The experiments are conducted for 802.11n at the 5 GHz band. The topology in all scenarios consists of one mAP, and two fSTAs connected through the backhaul

switch with the cloud. Beside to the orchestrador, the cloud hosts the applications accesses by the mobile element. The iperf3 tool is used to generate UDP traffic between the ends of the communication similar to those used in common real-time applications.

4.4 Proof-of-concept and evaluation

In this section, we aim to evaluate the impacts of our handover, and packet duplication approaches in the latency, throughput, and loss. We implemented a prototype in our testbed as a proof-of-concept to validate the proposed solution, and tools like iperf, ping, bwm-ng and tcpdump are used to collect and to analyze how our solution affects an E2E communication.

4.4.1 Orchestration for seamless and reliable handover

In this section, we aim to evaluate SDN-NFV solution for seamless and reliable handover, by using packet duplication approach. We evaluate the latency, throughput, and loss during a E2E transmission. We also implement a prototype in our testbed as a proof-of-concept to validate the proposed solution, and tools like iperf, ping, bwm-ng and tcpdump are used to collect and to analyze how our solution affects an E2E communication.

4.4.1.1 Experimental methodology and workload

The first objective is to run a simple experiment seeking to demonstrate the basic functionality of our architecture by reducing mobility interruption times during migration (i.e., when the mAP migrates from one fSTA to another). For this purpose, an experiment collects throughput information in an E2E communication that performs handover processes and packet duplication under our approach.

The second objective is to analyze our architecture performance with different mechanisms implemented. Performance metrics are evaluated by using services average latency in Round Trip Time (RTT) between the mobile element and the cloud, and packet loss improvement in the presence of lossy wireless channels. In this way, a second experiment was carried out to measure RTT and to analyze the latency in good wireless channel conditions i) without performing handovers (assumed as baseline), ii) performing simple handovers and iii) with handover implementing packet duplication during the transitions period. The experiment runs over 1 minute, sending a ping packet every 0.5 ms and the handover processes are triggered by the orchestrator every 1 second. This handover rate simulates a mobile element

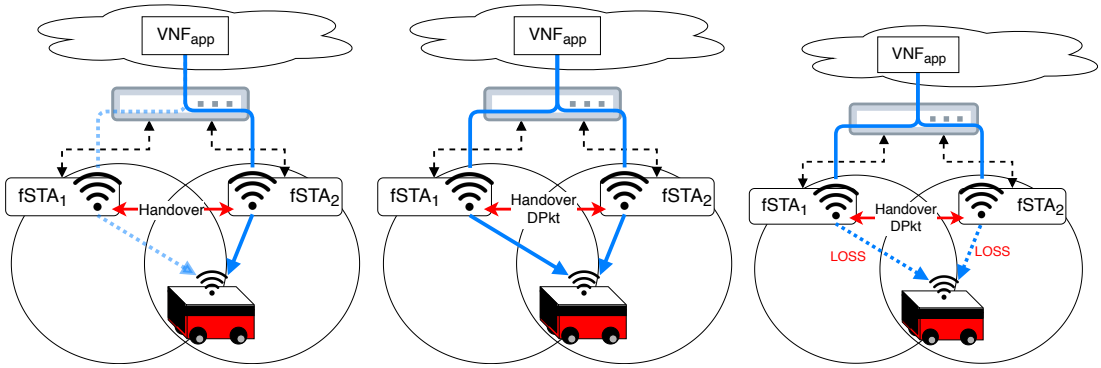


Figure 4.5: Scenarios: a)Throughput in seamless handovers: multi-connectivity with packet duplication from $t = 8s$ to $t = 18s$.

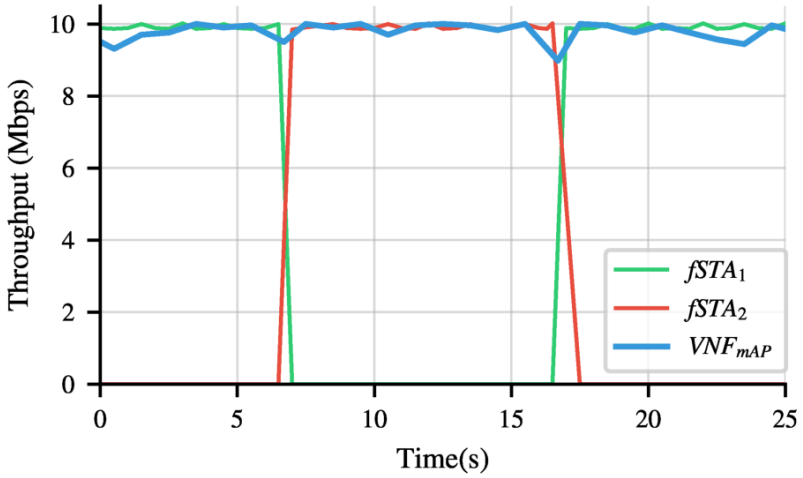


Figure 4.6: Throughput in seamless handovers: handover ($t = 7s$) and failover ($t = 18s$), without packet duplication.

that travels at a speed of 1 m/s in a small cell scenario with cell size of $1m^2$, where handover processes are very frequent and their performance is decisive for the communication reliability.

A third experiment seeks to analyze redundant transmissions by packet duplication. The experiment also runs over 1 minute in the presence of wireless channels with losses for both links (mAP-fSTA$_1$ and mAP-fSTA$_2$), so the orchestrator implements full packet duplication through both fSTAs.

4.4.1.2 Throughput in seamless handovers

In the first experiment, we measure the throughput from the VNF$_{app}$ in the Cloud to the mAP→VNF$_{app}$, the experiment duration was 25 seconds generating 10 Mbps UDP packets from the source. To understand the E2E throughput, a handover is

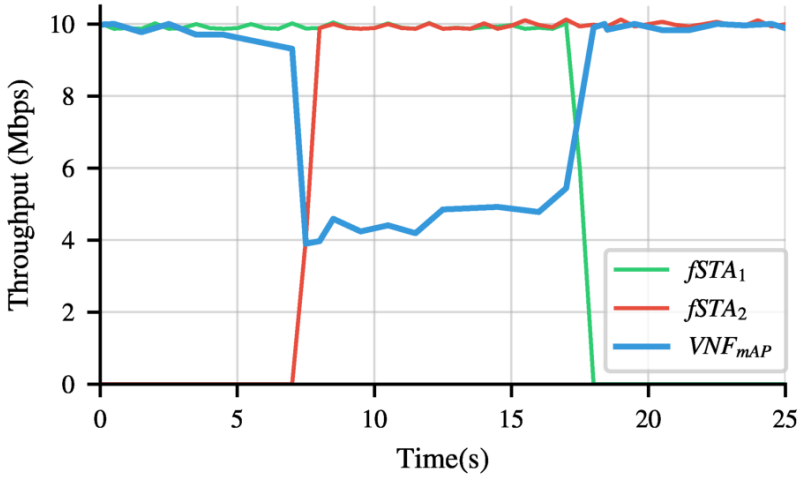


Figure 4.7: Throughput in seamless handovers: multi-connectivity with packet duplication from $t = 8s$ to $t = 18s$.

triggered from $fSTA_1$ to $fSTA_2$ at t = 7 s and then a fault is introduced in the $fSTA_2$ when t = 18 s. As it can be seen in Figure4.6, there is no interruption times due to mobility of the mAP. The system keeps the throughput performance, despite a little degradation around 5% during the handover and 10% during failure recovery. The difference can be explained by the reactive process in the failover process, since the orchestrator needs to wait for a notification of failure to move the traffic to another fSTA. Despite the performance degradation, there were only few packet losses during the experiment, not very representative for the final communication performance.

Also as part of the first experiment and under the same conditions, the packet duplication scheme implemented in the architecture was tested. In this case, the full duplication of the traffic is carried out through $fSTA_1$ and $fSTA_2$, as shown in Figure 4.7. A throughput degradation of 50% is expected as long as the duplication of traffic happens (blue line). This occurs due to the fact that at the destination, the mAP manages two data connections through the same wireless interface. This result evidences the existing trade-offs between reliability versus throughput or traffic density that can be managed by area. This requires an efficient orchestration scheme capable of determining the appropriate times to start/stop the duplication of packages, so that the best reliability-throughput trade-off can be obtained. For this purpose, some techniques can be shipped in the orchestrator, such as machine learning with supervised learning, determining the best moment to start duplicating packets fitting this trade-off. Then, a machine learning algorithm can classify accordingly depending on the application's requirements and the parameters exposed by the orchestrator.

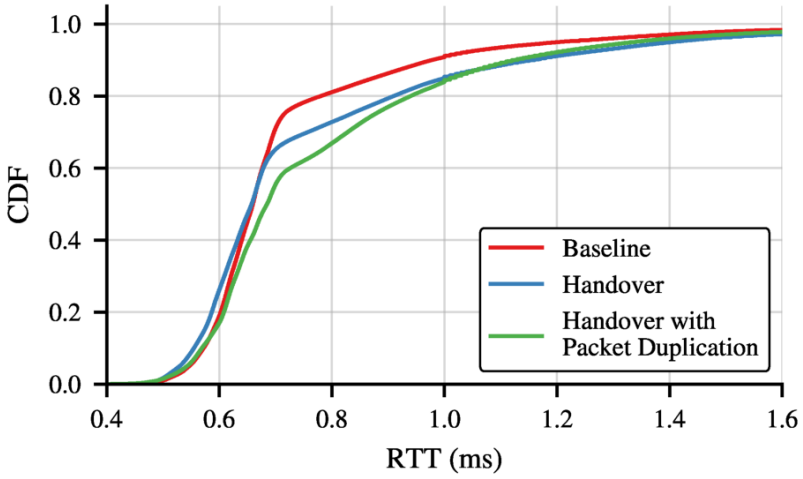


Figure 4.8: Comparison of latency during handovers: CDF of the RTT measured in each scenario.

4.4.2 Comparison of latency during handovers

The results of the second experiment related to latency are shown in Figure 4.8. The figure presents the end-to-end RTT CDF without performing handovers (baseline), performing simple handovers and with handovers implementing packet duplication during the transitions periods. The results show that for 60% of the packets, analyzed by the RTT, the handover scheme does not introduce noticeable latency when compared to the baseline. However, packet duplication adds extra latency due to de-duplication process carried out with a VNF scheme at user plane. For about 85% of the packets, the handover RTT already matches the duplication RTT, and both differ about 0.2 ms from the baseline, which represents the highest latency introduced by any of the schemes implemented in our architecture. Finally, 99% of the packets experience less than 2.0 ms of RTT in the three scenarios. This indicates that our architecture can even support the 1 ms latency real-time applications requirements described in URLLC.

4.4.3 Impact of packet duplication on packet loss

The last experiment analyzes the performance of our packet duplication solution under lossy channel conditions imposed by lowering transmitted powers. For this experiment, the validation consists in comparing the number of losses, sending in the same condition using: i) single path to $fSTA_1$ or $fSTA_2$; ii) multiple paths using $fSTA_1$ and $fSTA_2$. Figure 4.9 presents the packet loss percentage measured on each link.

The results show that packet duplication does significantly reduce the variability

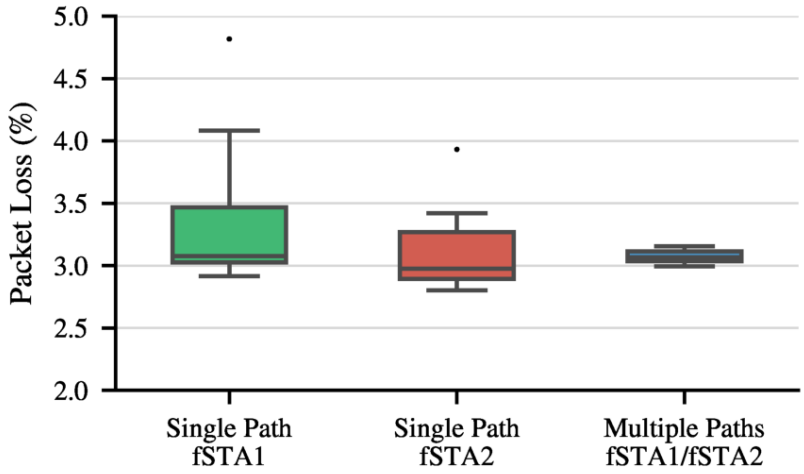


Figure 4.9: Impact of packet duplication on packet loss: with and without packet duplication.

of packet loss. However, the average packet seems to be unaffected due to the wireless channel used in this experiment. The main reason for this result is that the links used for parallel transmissions still have a high level of correlation, so the negative effects that affect one link would most likely affect the other. Thus, the identification and use of poorly correlated channels is a necessary condition to be tackled to increase the reliability.

4.5 Chapter Remarks

Reliability and latency will certainly be requirements for any URLLC offload solution using WiFi. The solution presented an architecture for better mobility management in WiFi networks by using orchestration of SDN-NFV functionalities. Our experimental results brought evidence that we can significantly reduce communication interruption caused by handover and failover events. Moreover, a packet duplication scheme over a multipath source-routing strategy does not introduce significant latency in E2E communication. Future work will involve efforts in the direction of WiFi-5G orchestration integration. Furthermore, by centralizing the channel state information of all wireless elements, a systematic reliability characterization and the support of the orchestrator prediction can be tackled by machine learning techniques, giving us more accurate information about the correlation of the different wireless channels. To this end, latency and jitter will be achieved to fully meet 3GPP by exploiting deterministic forwarding mechanisms for backhaul [Martinello et al. 2014], and datacenter networking [Liberato et al. 2018, Dominicini et al. 2017].

Chapter 5

Expressive and Agile Orchestration by Source Routing

5.1 Overview

The new generation of networks requires routing and forwarding packets under strict reliability requirements. For that purpose, the exploitation of redundant paths is one of the techniques that can deliver more reliability in network communications. The modern networks are characterized by topologies with high path diversity, which can be exploited to route traffic across diverse paths. Multipath routing can be a way to provide an efficient usage of the existing paths guaranteeing high-quality network services, and increasing the reliability and resilience in the end-to-end communications [Nielsen et al. 2018].

Multipath routing has emerged as a technology which can offer benefits for an expressive and agile orchestration as follows [Suer et al. 2020, Singh et al. 2015] :

- *Reliable communication* can be realized with the implicit fault tolerance aspect of multipath provisioning. In single path routing, when a path fails (e.g fiber cuts), routing protocols use alternate paths [Ramos et al. 2013]. The application is interrupted for the transitional time until an alternate path is set up between end-hosts. On the other hand, in the multipath scenario, mostly of the concurrent flows get affected, as they generally use disjoint paths. The lost packets can then be quickly retransmitted over existing non-faulty paths. Hence, the communication remain uninterrupted in the multipath scenario, albeit at lower throughput.

- *Traffic Engineering (TE)* : TE mechanisms map the traffic flows and network resources of a network in such a way that some of the major objectives, such as reliable communication, higher throughput, minimum delay, congestion control can be achieved ;

- *Load balancing and congestion control* : are important aspects of TE and can be achieved by using multipath provisioning. Traffic flows can be distributed over multiple concurrent paths such that all the links are optimally loaded, thereby avoiding network hot-spots. If some of the links/nodes are congested in the network, multipath routing can be efficiently used to shift fraction of the traffic from congested paths to less congested ones. In order to achieve load-balancing, either routers need to disseminate link-load information in the network, or the end-hosts.

- *Security* : While single-path routing is vulnerable to security threats, such as denial-of-service attack (by over-loading a particular node/link/path), multipath routing can provide greater security by dispersing data over multiple paths between end-hosts, where each path carries a portion of data between a source-destination pair. Moreover, multipath routing using unpredictable selection of links/paths makes it difficult for an attacker to conduct such attack against any single link/path [Lee et al. 2007].

In order to classify any multipath approach as basis for our source routing mechanism to guarantee an expressive and agile orchestration, there are three basic dimensions to consider [Suer et al. 2020, Singh et al. 2015]:

1. *Multipath computation algorithm* to compute multiple paths for a flow; To find multiple paths for a given traffic flow, multipath computation algorithms require a global view of the network topology as well as its resources. We may select paths that are *node-disjoint* (no common nodes except source and destination), *link-disjoint* (no common links) or *non-disjoint* (may have common nodes as well as links). Node/link-disjoint paths improve fault-tolerance and offer more aggregate bandwidth than non-disjoint paths, since a bottleneck node/link failure for non-disjoint paths can severely impact the performance of multiple paths. However, non-disjoint paths are easy to discover due to no constraint on common nodes and links.

2. *Multipath forwarding algorithm* to forward packets on diverse paths; Once intermediate routers and end-hosts compute the connectivity (path) information, the subsequent question that arises is: how to forward packets along these multiple paths? Forwarding is a method which maps incoming packets to outgoing links. In today's IP network, packets are forwarded along a shortest path by the intermediate routers based on their destination addresses. The well-known forwarding mechanisms that can be utilized in multipath packet forwarding are *destination-based (hop-by-hop) forwarding and source-based routing approach.*

3. *Traffic splitting algorithm* to effectively split traffic across multiple paths, aiming to support requirements such as reliable communication, congestion control, traffic engineering, load balancing and security. Once a set of paths for a given flow are determined and a forwarding technique is selected, the source node (edge) can begin sending data to the destination along the specified paths. Traffic splitting along multiple paths refers to how to distribute traffic over multiple paths. Broadly speaking, traffic splitting algorithms can be classified as *Round-Robin*, *Per-flow* , *Burst (or Flowlet)* with traffic splitting functions at the granularity of burst (i.e., a group of packets) and *Per-packet* traffic splitting techniques [Suer et al. 2020].

We organize the contributions of this chapter as follows:

- (i) we propose M-PolKA, a topology-agnostic RNS-based multipath SR scheme architecture (Section 5.2);
- (ii) we propose a technique to enable the implementation of the polynomial mod operation in P4-enabled programmable switches by reusing Cyclic Redundancy Check (CRC) hardware [Dominicini et al. 2020] (Section 5.3).
- To evaluate the aforementioned work (Sections 5.4 and 5.5),

  (i) we demonstrate in an emulated environment that M-PolKA can achieve similar performance to list-based approaches (Sections 5.4.1 and 5.5.1);

  (ii) we prototype and evaluate into SmartNIC to demonstrate that the M-PolKA can achieve similar performance to list-based approches (Sections 5.4.2 and 5.5.2);

  and (iii) we prototype and evaluate into P4-based ASIC Tofino switches[1]
- To demonstrate the feasibility of using polynomial operation in the new modern and high-capacity P4 programmable switches with the same performance comparing to list-based and table-based approaches (Sections 5.4.3, 5.5.3, and 5.5.4).
- In Section 5.6 we propose, demonstrate and evaluate, as a use case, a cross-layer network programmability with agile and expressive orchestration.

[1] https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series/tofino.html

5.2 Multipath Routing Proposal - MPolKA

A major challenge to support an efficient multipath routing is the inherent data and control planes scalability limitations in a legacy SDN table-based approach [Suer et al. 2020]. As the switching hardware supports limited table sizes from the data plane side, the growing number of table entries leads to scalability and performance issues. From the control plane side, path diversity brings the challenge of dealing with many control messages needed to reconfigure the network and consequently keep the multipath state. This is far from a trivial problem due to the scale, dynamics, heterogeneity, mobility, and high-performance required by modern applications [Guimaraes S et al. 2020].

In this context, *source routing (SR) schemes for multipath forwarding algorithm* arise as strong candidates to replace table-based routing since they avoid the reconfiguration of all the nodes along the path and to overcome data- and control-plane scalability limitations [Komolafe 2017]. These schemes allow traffic engineering to dynamically exploit all existing paths to achieve maximum throughput [Jyothi et al. 2015]. SR also reduces the control signaling and latency related to path setup convergence, once migrating paths are only a matter of changing the state at source or edge nodes[2], which are ingress and egress domain gateways.

In this chapter, we want to push to an extreme design choice and answer the following questions:

- Is it possible to define a fully stateless multipath SR approach (i.e., no state in the core nodes, nor in the packet) in a network that intrinsically enables

[2]The edge node may be a virtual switch in a server, a hypervisor, a top-of-rack (ToR) switch.

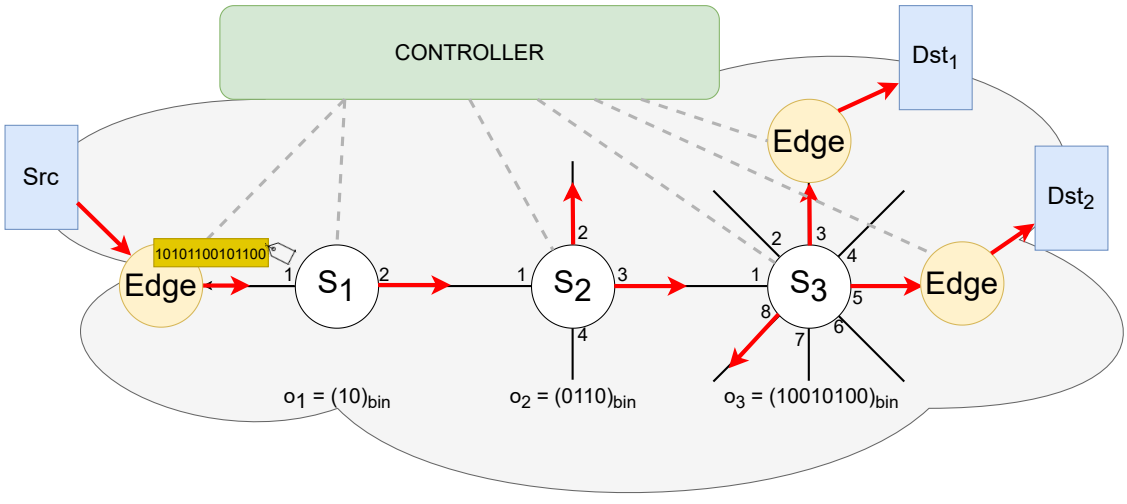


Figure 5.1: Example of source routing by using M-PolKA.

reliable communication via exploitation of path diversity ?

- How to implement such approach in modern and commodity network hardware with support to any topology?

To this end, we propose a SR multipath approach named **M-PolKA (M**ultipath **Pol**ynomial **K**ey-based **A**rchitecture), which explores special properties from the Residue Number System (RNS) with polynomial arithmetic using Galois field (GF) of order 2 [Shoup 2009], known as GF(2). These properties are topology-agnostic and guarantee that the node sequence is irrelevant to derive the routing label, which remains unchanged throughout all the path [Martinello et al. 2014, Wessing et al. 2002]. M-PolKA generalizes a previous work named PolKA [Dominicini et al. 2020], which only focused on providing an RNS-based SR scheme for single path.

In M-PolKA multipath routing shown in Figure 5.1, the architecture is composed of: (i) edge nodes, (ii) core nodes, and (iii) a SDN Controller, responsible for configuring core and edge nodes. The SR relies on three polynomial identifiers over GF(2): (i) *routeID*: a route identifier, calculated by the controller using the polynomial CRT and embedded into the packet by the edge nodes; (ii) *nodeID*: an identifier previously assigned to core nodes by the controller in a network configuration phase; and (iii) *portID*: an identifier assigned to the output ports of each core node.

5.2.1 M-PolKA usage example

Figure 5.1 shows a usage example in a topology composed of three core nodes. Suppose the host S wishes to communicate with destinations Dst_1 and Dst_2. Consider that packets should be routed via a selected path, represented by S_1, S_2, S_3 core nodes and their respective output ports.

In **M-PolKA** core nodes, the transmission states of the output ports are given by the remainder of the binary polynomial division (i.e., a $\mod$ operation) of the route identifier of the packet by the node identifier.

To understand all the steps on how M-PolKA works, let $S = \{s_1(t), s_2(t), \ldots, s_N(t)\}$ be the set of the polynomials representing the *nodeID*s of the nodes in this path. The set S must be composed of pairwise co-prime (irreducible) polynomials and satisfy the condition $deg(s_i) \geq nports$, where $nports$ denotes the number of ports in the node.

Let O = $\{o_1(t), o_2(t), \ldots, o_N(t)\}$ be the set of N polynomials, where $o_i(t)$ represents the transmission state of the ports (i.e., 0 does not transmit, 1 transmits a packet copy). For instance, the polynomial $o_i(t) = a_2t^2 + a_1t + a_0$ maps the state $a_2a_1a_0$, which means that there are 3 ports in the node s_i, and each coefficient represents the state of one port. If the output port polynomial is $o_i(t) = t^2 + t = 110$ then $a_2 = 1$,

$a_1 = 1$, and $a_0 = 0$. This means that port 2 is transmitting, port 1 is transmitting, and port 0 is not transmitting.

The degrees of the polynomials assigned to s_1, s_2, and s_3 must be equal or greater than 2, 4, and 8, respectively, to represent the necessary number of transmission states at each node. Consider that the following irreducible polynomials are assigned to the s_i nodes:

$$s_1(t) = t^2 + t + 1 = 111$$
$$s_2(t) = t^4 + t + 1 = 10011$$
$$s_3(t) = t^8 + t^4 + t^3 + t + 1 = 100011011$$

Considering the nodes s_1, s_2, and s_3 and the transmission states of Figure 5.1, the output ports set O is composed by:

$$o_1(t) = t = 10$$
$$o_2(t) = t^2 + t = 0110$$
$$o_3(t) = t^7 + t^4 + t^2 = 10010100$$

Thus, $R(t)$ must satisfy the conditions of Eq. 5.1:

$$R(t) \equiv (t) \mod (t^2 + t + 1)$$
$$R(t) \equiv (t^2 + t) \mod (t^4 + t + 1)$$
$$R(t) \equiv (t^7 + t^4 + t^2) \mod (t^8 + t^4 + t^3 + t + 1)$$

As in CRT for polynomials (Section 5.2.2), we have:

$$M(t) = (t^2 + t + 1) \cdot (t^4 + t + 1) \cdot (t^8 + t^4 + t^3 + t + 1)$$
$$m_1(t) = s_2(t) \cdot s_3(t) = (t^4 + t + 1) \cdot (t^8 + t^4 + t^3 + t + 1)$$
$$m_2(t) = s_1(t) \cdot s_3(t) = (t^2 + t + 1) \cdot (t^8 + t^4 + t^3 + t + 1)$$
$$m_3(t) = s_1(t) \cdot s_2(t) = (t^2 + t + 1) \cdot (t^4 + t + 1)$$

And solving Eq. (5.5), we find the polynomials $n_i(t)$:

$$n_1(t) = t, \quad n_2(t) = t^3 + t + 1, \quad n_3(t) = t^6 + t^5 + t^4$$

Finally, we can calculate $R(t)$ according to Eq. (5.2):

$$R(t) = t^{13} + t^{11} + t^9 + t^8 + t^5 + t^3 + t^2 = 10101100101100$$

Note that a polynomial $f(t) = a_n t^n + a_{n-1} t^{n-1} + \ldots + a_1 t^1 + a_0 t^0$ can be represented by the bit vector $a_n a_{n-1} \ldots a_1 a_0$. Thus, an identifier is represented by a bit vector formed by the coefficients of a polynomial, which are either 0 or 1. Also, the bit length of the identifier is $len(f)$.

Based on the definition of the path represented by S and O, the Controller calculates the *routeID* using the polynomial CRT (see Section 5.2.2) as the polynomial $R(t)$ that satisfies:

$$R(t) \equiv o_i(t) \mod s_i(t), \quad for \quad i = 1,2,\ldots N \tag{5.1}$$

The *routeID* is embedded in the packet by the edge element, and the forwarding operation in each core node calculates the transmission state of the output ports as the remainder of the euclidean division of the *routeID* in the packet by its *nodeID*: $o_i(t) = < R(t) >_{s_i(t)}$. Therefore, packets receive at the edge the *routeID* 10101100101100, then each node calculates the remainder of $R(t) = 10101100101100$ by its own *nodeID* $s_3(t) = 100011011$ which gives $o_2(t) = 100101000$. For example, in s_3 ports 3, 5 and 8 must transmit packets.

The Controller may proactively compute $R(t)$ or calculate it when the first packet of a flow arrives. On the other hand, core nodes only execute a simple $\mod$ operation per packet.

The following subsections are organized as follows: i) to describe the mathematical formulation and scheme supported by M-PolKA (5.2.2); and ii) scalability analysis (5.2.3).

5.2.2 Mathematical Background for M-PolKA approach

Polynomial Ring over GF(2): Let GF(2) = $\{0,1\}$ be the Galois Field of order 2, whose elements are residue classes modulo 2. The arithmetic operations of addition and multiplication in GF(2) are defined modulo 2. The set of all polynomials in one variable t with coefficients in GF(2), called polynomials over GF(2), is a ring considering the arithmetic operations of addition and multiplication modulo 2. If $f(t) = a_n t^n + a_{n-1} t^{n-1} + \ldots + a_1 t + a_0$ is a polynomial over GF(2), where $a_n \neq 0$, n is defined as the degree of $f(t)$, denoted by $deg(f)$. The length of $f(t)$, denoted by $len(f)$, is defined by $len(f) = deg(f) + 1$.

Euclidean Division Theorem for Polynomials: Let $f(t)$ and $g(t)$ be polynomials over GF(2) , where $g(t) \neq 0$. There exist unique polynomials $q(t)$ and $r(t)$ over GF(2) such that $f(t) = g(t).q(t) + r(t)$, where either $r(t) = 0$ or $deg(r) < deg(g)$. The polynomial $r(t)$ is called the remainder of the division of $f(t)$ by $g(t)$, and will be denoted by $< f(t) >_{g(t)}$.

Polynomial congruence: Given $f(t)$, $g(t)$, and $h(t)$ polynomials over GF(2), we

say that $f(t)$ is congruent to $h(t)$ modulo $g(t)$, and write $f(t) \equiv h(t)\ mod\ g(t)$, if $h(t) =< f(t) >_{g(t)}$.

Irreducible Polynomials: A non-zero polynomial $g(t)$, is called a divisor of $f(t)$ over GF(2) if $f(t) = a(t).g(t)$, for some polynomial $a(t)$ over GF(2). Two polynomials $f(t)$ and $g(t)$ over GF(2) are coprime if their only common divisor is 1. A non-constant polynomial $f(t)$ over GF(2) is called irreducible over GF(2) if its only divisors are possibly a constant polynomial and itself.

Chinese Remainder Theorem (CRT) for polynomials: Let $s_1(t), s_2(t), \ldots, s_N(t)$ be monic pairwise coprime polynomials over GF(2) and let $M(t) = \prod_{i=1}^{N} s_i(t)$. There exists a unique polynomial $R(t)$ over GF(2) with $deg(R) < deg(M)$, satisfying $R(t) \equiv o_i(t)\ mod\ s_i(t)$, for $i = 1,2,...,N$, where:

$$R(t) = < \tilde{R}(t) >_{M(t)} \tag{5.2}$$

$$\tilde{R}(t) = \sum_{i=1}^{N} o_i(t) \cdot m_i(t) \cdot n_i(t) \tag{5.3}$$

$$m_i(t) = M(t)/s_i(t) \tag{5.4}$$

$$n_i(t) \cdot m_i(t) \equiv 1 \quad mod \quad s_i(t) \tag{5.5}$$

The computation of $n_i(t)$ can be implemented by using the Extended Euclidean Algorithm, which basically consists in applying the Euclidean Division Theorem several times. The algorithm complexity for computing $R(t)$ is $\mathcal{O}\big(len(M)^2\big)$ [Shoup 2009].

5.2.3 Scalability analysis

The goal of this section is to investigate the overhead of our new scheme in the bit length of *routeID* in comparison to the Port Switching and integer RNS-based approaches. The bit length of $R(t)$, $len(R)$, in PolKA is given by the equation:

$$len(R) = len(< \tilde{R}(t) >_{M(t)}) \leq \sum_{i=1}^{N} deg(s_i) \tag{5.6}$$

Algorithm 1 shows a pseudo-code for computing the maximum $len(R)$, given: the number of ports in each node ($nports$), the number of nodes ($size$), and the topology diameter ($diameter$). For the sake of simplicity, we consider that all nodes have the same number of ports. A list of *nodeID* polynomials ($nodelist$) is generated, which consists of $size$ irreducible polynomials with degree greater than or equal to the minimum degree ($mindeg$). Note that we select polynomials with the lowest possible degree (e.g., if $mindeg = 5$, we start assigning one of the 6 existing irreducible polynomials of degree 5 to nodes, and, if necessary, we use the 9 existing irreducible

Algorithm 1 Computation of the maximum $len(R)$.

```
1: function maxlen(nports, diameter, size)
2:     mindeg ← nports
3:     nodelist ← generate_irred_poly_list(mindeg,size)
4:     pathlist ← get_last_itens(nodelist,diameter)
5:     length ← 0
6:     for elem ∈ pathlist do
7:         length ← length + deg(elem);
8:     end for
9:     return length Maximum len(R)
10: end function
```

polynomials of degree 6, and so forth). Thus, $nodelist$ is already ordered by degree. Finally, we select the worst case scenario in which the polynomials in $nodelist$ with the greatest degrees assigned to the nodes in the longest possible path (i.e., the diameter). To this end, we pick the x last elements of $nodelist$, where $x = diameter$, and calculate the maximum $len(R)$ according to Eq.(5.6).

Table 5.1 compares the scalability of M-PolKA, ELMO [Shahbaz et al. 2019], and BIER [Wijnands et al. 2017] for two common DC topologies (fat-tree and two-tier), and three continental backbone topologies [Routray et al. 2013] (ARPANET, and GEANT2). This topology set covers diverse properties for switch fan-outs (i.e. number of ports), network diameter and size. For backbone topologies, as the number of ports varies per node, we considered $nports$ as the maximum number of ports of any node. The results for the integer RNS-based approach were omitted, because they are very close to M-PolKA. This means that using M-PolKA instead of the integer version does not incur in reserving extra bits for the *routeID* header.

For the topologies under worst case analysis in Table 5.1, the maximum $len(R)$ results show that M-PolKA fits existing packet headers (e.g., 96 bits of Ethernet source and destination addresses, or a stack of MPLS labels with 20 bits per label, plus BIER header). When deploying RNS-based SR, the cost to exploit RNS features may be to reserve more bits in the packet header for representing the *routeID*. Nevertheless, there are known techniques that make optimal assignment of *nodeID*s (avoiding the worst-case scenario), and reduce *routeID* length [Ren et al. 2017, Liberato et al. 2018].

As shown in Table 5.1, ELMO as a well-known existing multipath LB-SR, does not scale well with different topologies and demonstrates a good scale for only fat-tree topologies. On the other hand, BIER has a problem in scaling for topologies with higher diameters, such as fat-tree with either 16 or 24 pods. However, noteworthy to say that the number of bits needed by the BIER considering it as a hybrid SR (i.e., hop-by-hop match-action table operation), there is still the need to maintain the forwarding states on all nodes. It penalizes either the convergence time, once we

Table 5.1: Scalability of Header sizes (in bits) for different configurations.

Topology	$nports$	$diam.$	$size$	**Bits for M-PolKA**	**Bits for ELMO**	**Bits for BIER**
Two-tier S6 L6*	24	3	12	72	**74**	12
Two-tier S12 L12*	24	3	24	72	**122**	24
Two-tier S16 L16*	24	3	32	72	**154**	32
Fat-tree 8 pods	8	5	80	45	42	**80**
Fat-tree 16 pods	16	5	320	80	**82**	**320**
Fat-tree 24 pods	24	5	720	120	**122**	**720**
ARPANET	4	7	20	44	x	20
GEANT2	8	7	30	56	x	30

** Two-tier topology with 16 spine switches and 16 leaf switches.*

need to update the forwarding state on all nodes, or it becomes tough to explore other paths for a given E2E communication (i.e., for load-balance or protection paths).

5.3 Implementation Design

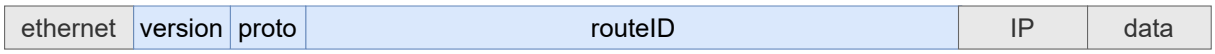
The most traditional way of executing source routing is a list-based SR (LB-SR) approach, which is the case of ELMO. Indeed, the routing label represents an ordered list (or stack) of output ports, and the forwarding operation is a pop of the first element [Jin et al. 2016]. Although this approach drastically reduces the burden of managing network states by eliminating tables in core nodes, it still needs to maintain a state in the packet by using a routing label *rewrite operation in every node* to update the list. This operation may be costly for packet networks and difficult to implement in other network domains [Wessing et al. 2002].

We have decided to extend the simplest LB-SR called Sourcey [Jin et al. 2016] by adding the support for multipath and multicast, in which it does not affect the properties of the operations needed to be performed in the list hop-by-hop. In reality, the difficulty of reproducing ELMO is inherent to the fact that ELMO only shares an initial P_4[3] code missing some critical information and code about the externs used in the code and also information on how to reproduce a new experiment minimally.

This section studies how to implement M-PolKA and simplified LB-SR methods according to P4 architecture [P4 v16 2020]. We decided to implement a simplified LB-SR with the multipath capability to make a baseline comparison with M-PolKA approach. Design choices may change according to application requirements and

[3] https://github.com/Elmo-MCast/p4-programs

(a) LB-SR header with variable length



(b) M-PolKA header with fixed length

Figure 5.2: LB-SR and M-PolKA headers.

target platform features. For instance, the *routeID* can be added in a new or existing header, while the header's size depends on the network topology, diameter, and the number of ports per switch. This section is organized as follows: i) Pipeline design (5.3.1); ii) Data plane (5.3.2); iii) Control plane (5.3.3).

5.3.1 Forwarding packets on multiple paths in P4 Pipelines

5.3.1.1 LB-SR pipeline

Our implementation of LB-SR creates a new header that includes the port stack after the Ethernet header, as shown in Figure 5.2(a). Each item of the stack has a `bos` (bottom of stack) bit and a port number. The `bos` bit is 1 only for the last entry.

When the packet reaches an edge node from an end-host, the `etherType` in the Ethernet header is TYPE_IPv4=`0x800`. Thus, the switch must parse the IPv4 header, encapsulate the SR header, and change the `etherType` to TYPE_SR=`0x1234`.

Each edge switch has a table, which is populated by the controller and maps destination IP address to their routing paths, represented by a port list. The result of this table lookup is an action that sets the output port to the directly connected core switch and encapsulates n SR headers, where n is the number of core switches in the path. At core switches, the pipeline of Figure 5.3(a) is executed. Firstly, the Ethernet header is parsed to check if the `etherType` is TYPE_SR. Then, the switch parses the first SR header, gets the version, which identifies the protocol version, and the bit vector (`bit_vector`). Finally, the switch pops this bit vector header from the stack. If the `bos` bit is one, it is identified as the last hop and then the `etherType` is changed to TYPE_IPv4. Another possible implementation for progressing in the list is to increment an index of each hop's current position. The protocol version identifies how the bit vector will be interpreted (i.e., version equal 1 is unicast or version equal 2 is multicast).

5.3.1.2 M-PolKA pipeline

The M-PolKA header contains a *routeID*, whose maximum length depends on the network topology, as explained in Section 5.2.3. Figure 5.2(b) shows the format of

(a) LB-SR (b) M-PolKA

Figure 5.3: Forwarding packets on multiple paths in P4 pipelines for core switches

M-PolKA header with a fixed length field for storing the *routeID*, after the Ethernet header. At edge switches, M-PolKA pipeline for encapsulating the SR header is similar to LB-SR, but the result of the table lookup is an action that sets the output port to the directly connected core switch and encapsulates a single *routeID*.

At core switches, the pipeline of Figure 5.3(b) is executed. When `etherType` is TYPE_SR, as M-PolKA only needs read access to packet headers, it uses the `lookahead` method of P4 language that evaluates a set of bits from the input packet without advancing the packet index pointer. To obtain the bit vector, which can represent either the output port or a transmission state (`t_state`), the switch has to perform a $\mod$ operation between the *routeID* in the packet and its own *nodeID*. To iterate onto the bit vector, we use a chain of `if (<condition>)` statements for each port. We do not use `resubmit` or `recirculate` actions once we observed a huge overload in the latency when we have used them since these built-in actions behave as repetition statements. The protocol version is only used to identify how the bit vector will be interpreted. In this case, when the version is equal to 1, the

packet is handled as unicast, whereas for a version equal to 2, the packet is handled as multicast.

Besides, in M-PolKA, there is no information in the header to identify the last hop. This is not a problem in networks, because the packet delivery to an edge switch represents the end of the SR path. At the edge switch, the SR header is removed and the etherType is changed to TYPE_IPv4.

5.3.1.3 Comparison between LB-SR and M-PolKA pipelines.

- LB-SR header has variable size, depending on the number of remaining hops in the path. On the other hand, M-PolKA has a fixed-length header to store the *routeID*;
- In P4, LB-SR needs to create one encapsulating action for each stack size (e.g., `add_header_1hop`, `add_header_2hops`, ...), which increases the number of code lines and memory for deploying the edge code;
- In LB-SR, each core switch performs a header rewrite to update the stack when performing the pop operation (yellow in Figure 5.3(a)). On the other hand, in M-PolKA (Figure 5.3(b)), the packet remains unchanged along the path;
- In LB-SR, the bit vector is directly available in the SR header, while M-PolKA requires an arithmetic operation over the *routeID* to calculate the bit vector.
- After acquiring the bit vector, for the multicast approach, the data plane iterates onto the bit vector in both approaches to check if the ports belong to the bit vector. If it is true, the data plane clones the packet that is enqueued in the packet buffer with the respective egress port to afterwards be processed by the egress stage. For the unicast approach, the data plane does the casting from the bit vector to an integer value, which identifies the output port.

Thus, if we can perform the $\mod$ operation of M-PolKA with equivalent latency to the rewrite operation of LB-SR, we could take advantage of RNS properties without compromising performance when compared to Port Switching.

5.3.2 Data plane

5.3.2.1 Reuse of CRC for implementing mod operation in P4

Polynomial or integer $\mod$ operations with non-constant operands are not natively supported by commodity network hardware and are not available in P4 language. Thus, as stated before, one of the main contributions of this chapter is to generalize

the technique proposed by [Dominicini et al. 2020], which allows the execution of the polynomial $\mod$ in hardware by reusing common CRC operations.

In CRC operations, sender and receiver agree on a generator polynomial (G), which is an $r+1$ bit pattern used for error-detection. For data D with d bits, the sender calculates additional r bits (R), and appends them to D in such a way that the result is a polynomial with $d+r$ bits that is divisible by G using modulo-2 arithmetic [Peterson and Brown 1961].

Thus, the CRC code is the remainder of D shifted left by r bits, divided by G: $R =< D \cdot 2^r >_G$. Therefore, we could try to map the *routeID* as D, and the *nodeID* as G. However, M-PolKA does not perform a shift operation over the *routeID* as done in the CRC strategy.

As the degree of G is r, this problem can be solved if we separate the *routeID* in two parts: $routeID = D * 2^r + dif$, where dif is the r least significant bits of the *routeID*. Firstly, we shift right the *routeID* by r bits to produce the data D, which will be the input of the CRC function. Then, the bits that were lost with the shift right operation (dif) can be added back to the calculated CRC remainder in the end of the computation to produce a bit vector. Since the degree of the bitmap obtained is less than r, the unicity property in division algorithm for polynomials assures that the outcome polynomial coincides with the remainder obtained by direct division of *routeID* by the *nodeID*. Also, in binary arithmetic, both the addition and subtraction operations are identical to the logical XOR operation. These steps are described as follows:

1. $G =$ *nodeID*, $r = degree(G)$
2. $D =$ *routeID* $\div\, 2^r$ ***(SHIFT RIGHT)***
3. $dif =$ *routeID* $- D * 2^r$ ***(SHIFT LEFT, XOR)***
4. $R =< D * 2^r >_G$ ***(CRC)***
5. $bit\_vectorID = dif + R$ ***(XOR)***

Therefore, the switch calculates the transmission state by using two *SHIFT*, one CRC, and two *XOR* operations, which is more computationally efficient than executing a division.

With the use of this technique, M-PolKA can be deployed in P4 targets that allow the configuration of generator polynomials. Since P4 supports CRC operations through the use of external libraries, called externs [P4 v16 2020], the support for customized polynomials depends on the architecture models and how specific

targets implement them. The PSA[4] and v1model[5] architectures support customized polynomials of 16 and 32 bits. In terms of targets, the software switch bmv2[6] and the hardware switch Tofino from Barefoot support customized CRC polynomials, while Netronome SmartNICs only support fixed CRC polynomials.

5.3.2.2 Transmission State

Let a bit vector representing the transmission state denoted by t_state_x. The t_state_x expresses the transmission state to one or many ports in a given hop, looking from the least significant bit (LSB) to the most significant bit (MSB). In this case, the data plane needs to perform the cloning/mirroring function, supported by PSA and v1model architectures, to steer the packet's clones to other ports. For instance, if we have a $t\_state_x = 100100$, the packet is steered to port 3 (LSB), and also a clone is steered to port 6 (MSB). To check if $\forall P_i \in t\_state = \{1,...,n\}$, where P_i represents a given port number and n represents the number of ports, we use the following steps:

1. $aux = portNumber - 1$ ***(SUBTRACTION)***
2. $mask = 1 \ll aux$ ***(SHIFT LEFT)***
3. $R = t\_state \wedge mask$ ***(AND)***

Hence, if $R > 0$ then the packet must be steered to the port P_i. The Code 5.1 shows our P4 implementation based on the aforementioned steps.

```
apply {
    /* ... */
    if (meta.version == 2){
        // to process the transmission state
        if(meta.count == 0){
            meta.count = 1;
            meta.has_lsb = 0;
        }
        // Porta 1
        if((meta.bitvector & (64w1 << (bit<63>)(meta.count - 1))) > 0){
            if(meta.has_lsb){
                clone_packet((bit<32>)meta.count);
            } else {
                meta.lsb_port = meta.count;
                meta.has_lsb = 1;
            }
        }
```

[4] https://p4.org/p4-spec/docs/PSA.html
[5] https://github.com/p4lang/p4c/blob/master/p4include/v1model.p4
[6] https://github.com/p4lang/behavioral-model

```
        meta.count = meta.count + 1;
        // End Port 1
    /* ... */
```

Code 5.1: Tranmission state P4 algorithm

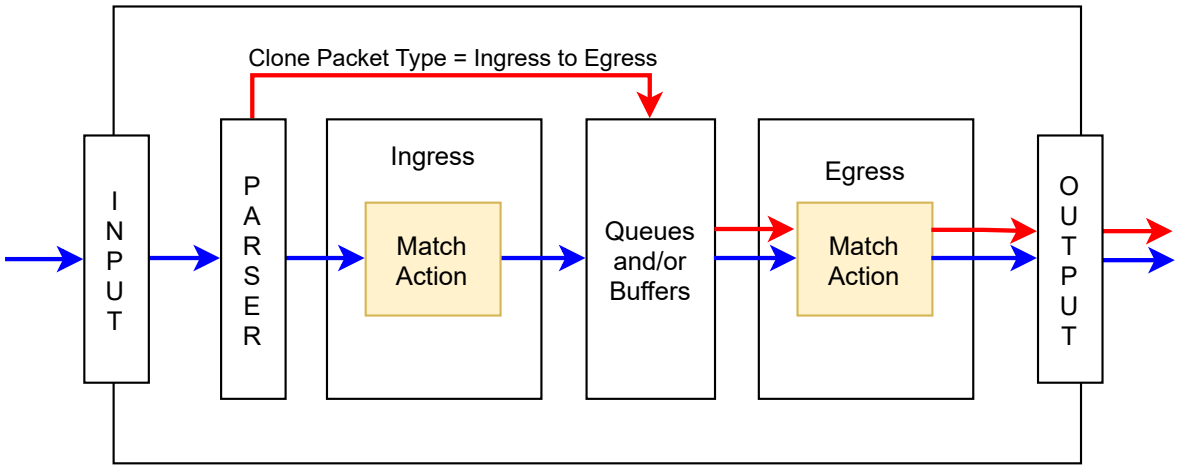


Figure 5.4: Cloning from Ingress to Egress. Adapted from [P4 2017]

The cloning/mirroring operation is a mechanism to create a copy of a packet and send the copy to a specified port. This mechanism is implemented by most of the targets with the support of two packet paths for packet cloning/mirroring: i) Ingress to Egress (I2E); ii) Egress to Egress (E2E) [P4 2018]. As shown by Figure 5.4, when the packet arrives at the Ingress stage, the switch performs the I2E clone function, and then a replica is sent to the beginning of the egress pipeline after passing through the buffer. To create a new clone instance, we must associate the packet to a given CloneSessionID. This specifies one of several possible clone session settings that were already pre-defined by the control plane (i.e., we can create many clones of one packet by using different CloneSessionIDs). For each clone session, we have to set two attributes:

1. *CloneSessionID*: 32 bits unique number.

2. *EgressPort*: 9 bits egress port number.

In the Code 5.2 we create the action ***clone_packet*** that must receive CloneSessionID parameter. Once this function is invoked, it calls the *clone* extern presented at the v1model with the parameters *CloneType.I2E* (Ingress to Egress) and *CloneSessionID*.

```
action clone_packet(bit<32> clone_session_id) {
    // Clone from ingress to egress pipeline
    clone(CloneType.I2E, clone_session_id);
}
```

```
apply {
    /* ... */
    clone_packet((bit<32>)meta.count);
    /* ... */
}
```

Code 5.2: Clonning/Mirroring Code

5.3.3 Control Plane

One of the M-PolKA control planes' main roles is topology discovery and monitoring since other steps depend on full knowledge of the network. The SDN Controller is also responsible for a network setup phase, where it assigns unique nodeIDs to each core node. With the knowledge of the number of nodes and the number of ports of each node, the controller must calculate the N irreducible polynomials for the nodeIDs while ensuring the degree. We have developed a Python library[7] to automatically calculate and return the nodeIDs based on a given topology specified via a JSON message, as shown in Code 5.3 one. The library abstracts the whole process in figuring out the irreducible polynomials and, based on that, the routeID calculation for a desired end-to-end communication (i.e., GF(2) arithmetic operations).

```
{ "name": "Topo01",
  "multicast_enabled": true,
  "nodes": [
    {
      "name": "switch 1",
      "type": 1,
      "ports": [1,2,3]
    },
    {
      "name": "edge 1",
      "type": 2,
      "ports": [1,2]
    },
    # ... #
  ],
  "links": [
      {
        "src": {
          "name": "s1",
          "port": 3,
        },
        "dst": {
          "name": "s2",
```

[7] https://pypi.org/project/polka-routing/

```
          "port": 2
        }
      },
      # ... #
    ]
  },
  "routes": [
    {
      "from": "e1",
      "to": "e2"
    },
    {
      "from": "e2",
      "to": "e1"
    },
  ]
}
```

Code 5.3: Example of a JSON message used in M-PolKA Python Library to get automatically irreducible polynomials and routeIDs

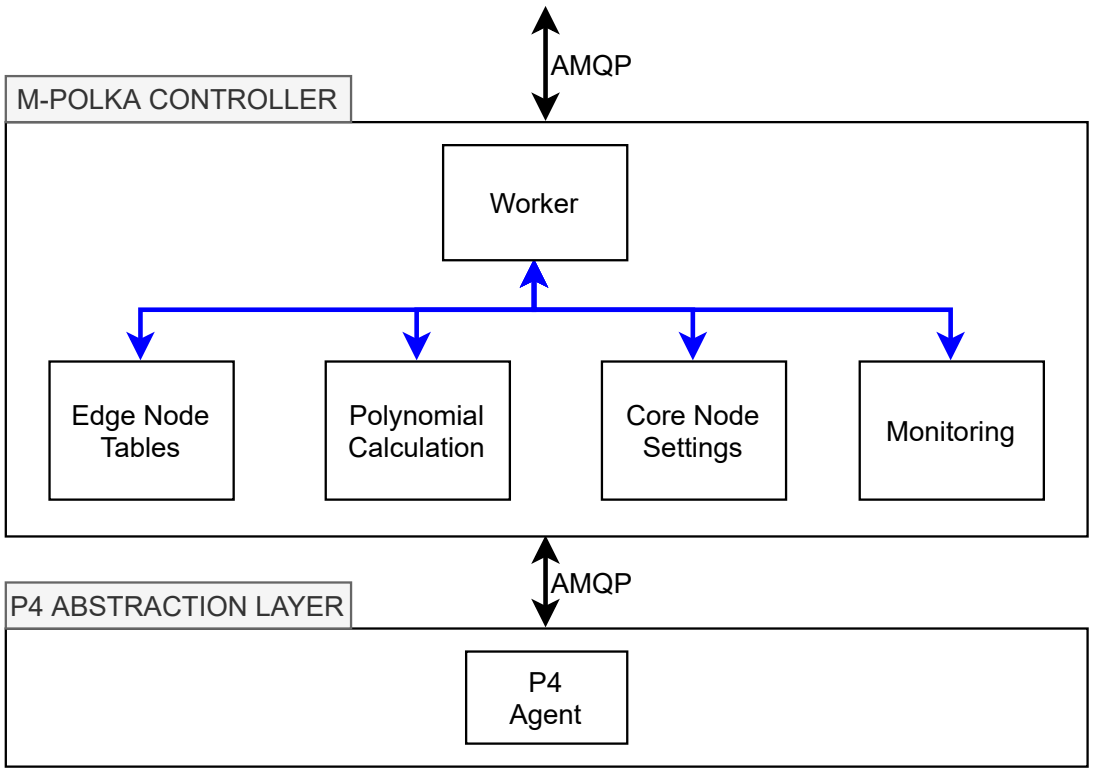


Figure 5.5: M-PolKA Controller and P4 Agent

As Figure 5.6 illustrates, the key components of the M-PolKA control plane architecture consist of:

- Edge Node Tables: responsible for inserting, deleting, and updating rules at

the edge nodes. Each rule in the table identifies the routeID used for a given flow, making it possible for the edge nodes to add and remove the appropriately routeID for each packet.

- Polynomial Calculation: calculate routeID around its domain to alleviate the orchestration overload by taking autonomous decisions regarding link failure, elements mobility, and the use of protection paths.

- Core Node Settings: in charge of setting nodeIDs and CloneSessionIDs up. This abstraction hides all the procedures involved in the setup of those parameters independently of the architecture. We have developed "drivers" to enable compatibility with both BMv2 and ASIC Tofino architectures. For the ASIC Tofino, we have developed a driver by using the proprietary Python Library provided by Inte/Barefoot that interacts with the switch device.

- Monitoring: collects and reports the network state, such as link failure and congestion, in order to provide a feedback loop needed for upper layer decisions (e.g., to a multi-domain orchestration layer).

- Worker: acts as an internal broker aiming to coordinate disaggregate controller applications with their actuation scope. It provides the opportunity to plug easily other third-party library functions or modifies existing functionalities via a unified interface.

P4 Abstraction Layer consists of a P4 agent that receives messages via a publish-subscribe channel and translates them to a P4 runtime command. Therefore, the P4 agent uses the same publish-subscribe channel to send some events regarding link failure and congestion to the M-PolKA control plane. The publish-subscribe paradigm provides more scalability, resilience, and maintainability because, in general, it offers a more loosely coupled communication system than gRPC[8] and point-to-point messaging technologies [Comer and Rastegarnia 2019].

5.4 Proof-of-concept and experimental validation

To evaluate the main functionalities of M-PolKA in comparison to the LB-SR, two prototypes were implemented: (i) an emulated setup to evaluate end-to-end scenarios, and (ii) a physical setup that uses Netronome SmartNICs[9] to evaluate forwarding in a single hop scenario.

[8] https://grpc.io/
[9] https://www.netronome.com/

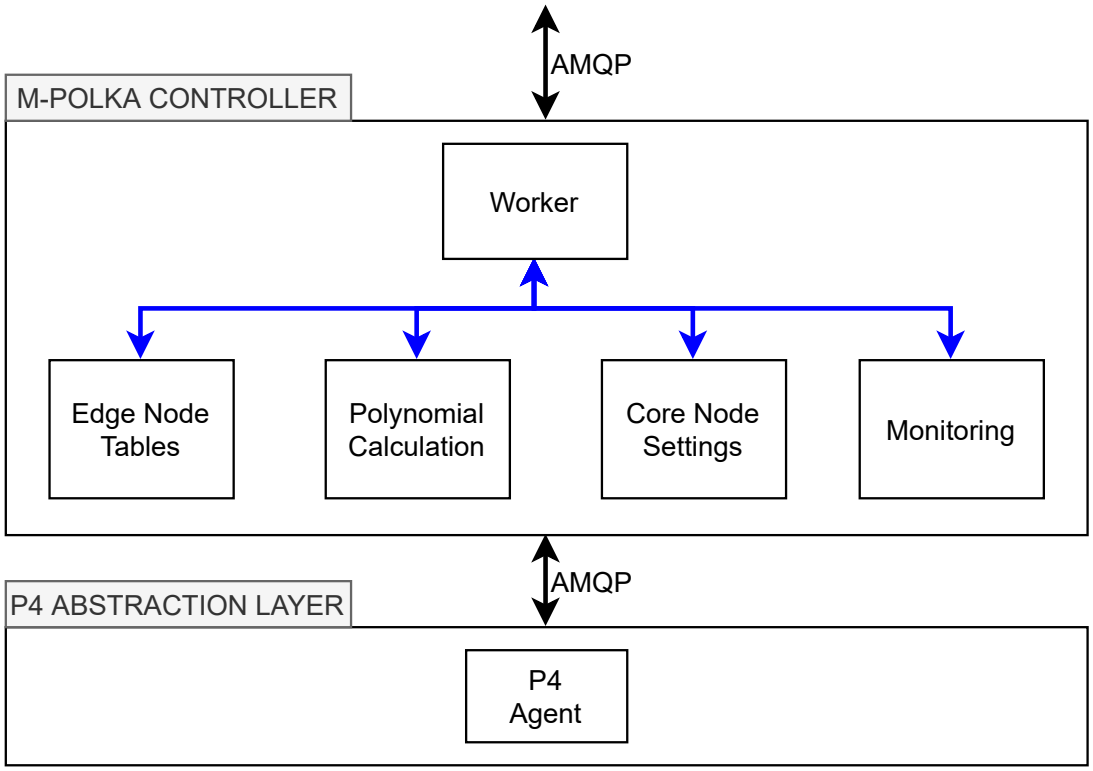


Figure 5.6: M-PolKA Controller and P4 Agent

5.4.1 Emulated M-PolKA prototype

5.4.1.1 P4 architecture and target

The software switch `bmv2 simple_switch` with the `v1model` architecture was selected as the target for this prototype, once it supports all the functionalities required by M-PolKA, such as the configuration of CRC polynomials. Also, we use the `clone3` primitive action (P4_16) to enable the multipath feature. The number of clone session IDs is limited to 65535 in the `bmv2 simple_switch`. It is important to highlight that this software switch is a user-space implementation that focuses on feature testing. There are other high performance implementations of P4 software switches and compilers (e.g., PISCES[10], P4ELTE[11], and MACSAD[12]), but they do not yet cover all the features required by M-PolKA. As these implementations arise, it will be possible to test our prototype with higher loads. For the time being, the solution was to limit the link rates to 10Mbps in our emulated prototype to avoid reaching the processing capacity limits of `bmv2 simple_switch`.

[10] http://pisces.cs.princeton.edu/
[11] http://p4.elte.hu/
[12] https://github.com/intrig-unicamp/macsad/

5.4.1.2 Setup description

The setup consists of one server Dell PowerEdge T430, with one Intel Xeon E5-2620 v3 2.40 GHz processor and 64 GB of RAM. To build our emulated environment, we used Mininet [Mininet 2018] with P4[13], which becomes a tool that augments the well-known Mininet emulator by including virtual wireless stations (STA) and access points (AP), and, likewise, allows running P4 programs on different switches. This functionality is crucial to emulate wireless and wired networks, which require different P4 programs for the edge, core, and AP elements.

5.4.1.3 Control plane implementation

It has two main functionalities: (i) for core: compute *nodeID*s, and configure core switches with their respective identifiers; and (ii) for the edge: compute routing paths for the traffic flows, calculate *routeID*s for these paths, and configure table entries in edge switches that will be responsible for encapsulating *routeID*s. For the RNS computation of *nodeID*s and *routeID*s, as described in Section 5.2.1, we developed a Python library presented at Section 5.3.3 that uses the package `galoistools` of the `sympy` library[14] for GF(2) arithmetic operations. For programming each core and edge switch, we developed a driver at the P4 abstraction layer in the control plane, as described in Section 5.3.3, which communicates with the switches by using the CLI commands provided by the `bmv2 simple_switch`. As shown in Figure 5.6, the control messages format is defined by an API that connects to the P4 abstraction layer via the AMQP interface.

5.4.1.4 Header size

The `bmv2 simple_switch` only supports the specification of CRC polynomials of 16 and 32 bits. Therefore, although our tests could use much smaller degrees, our PoC must adopt polynomials of degree 16. As the diameters of our test topologies are smaller than 10, the size of the M-PolKA header was defined as 160 bits. To a fairness comparison, we defined the LB-SR header as 16 bits (bos and port); therefore, for 10 hops, the array of headers has 160 bits.

5.4.2 M-PolKA deployment in SmartNICs hardware

In the emulated prototype, the CRC operation is executed in software by using CRC tables. However, the main benefit of using CRC is the execution of the $\mod$

[13] https://github.com/jafingerhut/p4-guide
[14] https://www.sympy.org/

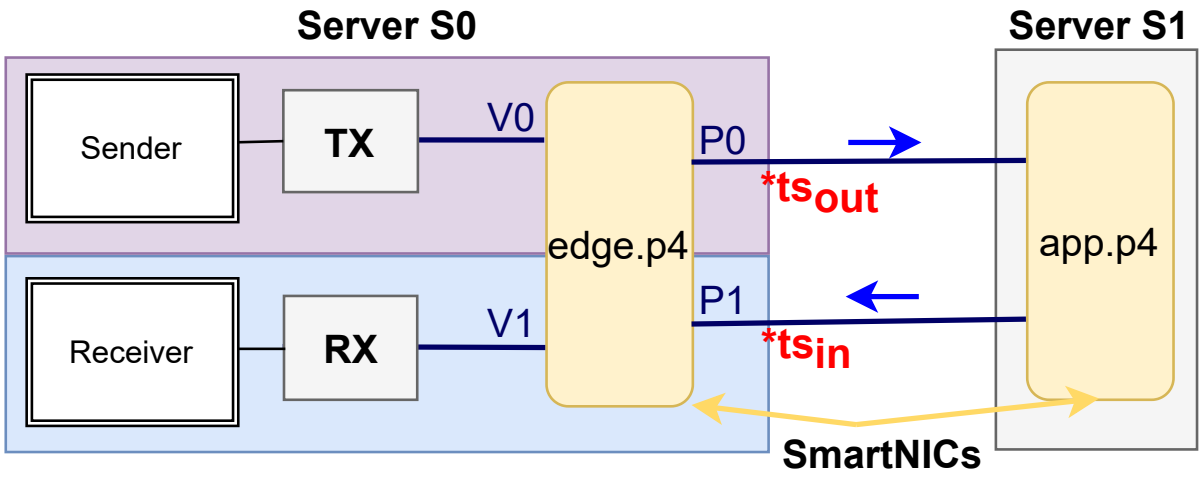


Figure 5.7: SmartNIC setup.

operation in hardware with better performance. To this end, we built a hardware prototype by using SmartNICs.

5.4.2.1 Setup description

The setup is illustrated by Figure 5.7 and consists of two servers: (i) S0: a device under test (DUT), running the core functionalities; and (ii) S1: a traffic generator (TG), running the edge functionalities, and transmitter (TX) and receiver (RX) functionalities in separate network namespaces. Both servers are Dell PowerEdge T430, with one Intel Xeon E5-2620 v3 2.40 GHz processor, 16 GB of RAM, and one Netronome Agilio CX 2x10GbE SmartNIC.

Netronome SmartNICs give access to hardware timestamps into the P4 programs.They partially implement the functionalities of *v1model* for P4 16, but it currently supports only a small set of fixed CRC polynomials, which restricts the use of its CRC hardware in M-PolKA. Nevertheless, we could use them for measuring forwarding latency in one core node, as proposed in the next section.

5.4.2.2 P4 programs

The P4 programs are adaptations of the codes used in the emulated prototype. The new edge code encapsulates both SR headers and a timestamp header for executing latency measurements, which is composed of an egress (ts_{out}) and an ingress timestamp (ts_{in}).

At server S0, packets are generated at TX and forwarded to the SmartNIC, where the edge adds SR (M-PolKA or LB-SR) and timestamp headers. When the packet leaves the edge, the hardware timestamp is assigned to ts_{out}. Then, it is transmitted to server S1 and processed by the core code at the SmartNIC, which is responsible

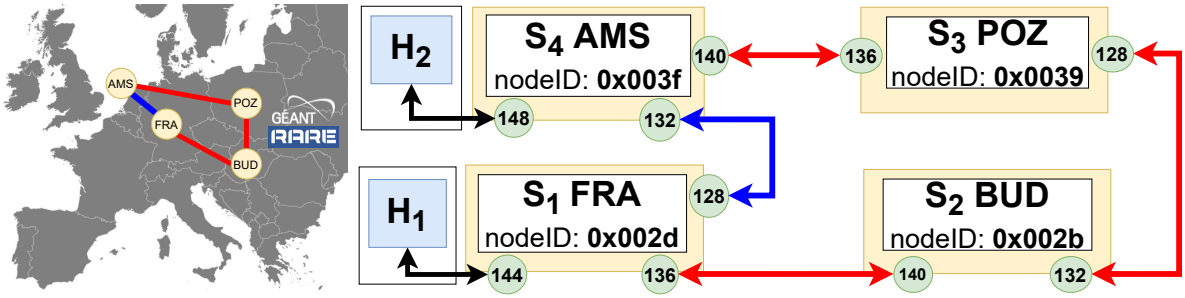


Figure 5.8: RARE/GÉANT P4 Lab European testbed [GÉANT 2021].

for parsing the SR header and computing the output port. Since it is impossible to configure customized CRC polynomials in the SmartNICs, we use a standard CRC operation with a fixed CRC-16 polynomial. In this way, we execute all the M-PolKA pipeline steps (including the CRC operation) to measure their contribution to the total latency. As a result, the packet is delivered to server S0, where the edge code assigns the value of the hardware timestamp to ts_{in}, removes the SR header, and delivers the packet to RX. Finally, all packets are captured with `tcpdump` tool, and parsed offline to extract the core forwarding latency for each packet by using $ts_{in} - ts_{out}$.

5.4.3 M-PolKA deployment in Tofino Switch

To validate and push the use of Cyclic Redundancy Check (CRC) to the production level, we propose an implementation that suits commercial programmable switches to prove the effectiveness of using it in a real-world environment. Therefore, i) we implement M-PolKA by using the P4 language [P4 v16 2020] in the high-performance switching ASIC Tofino; and ii) we deploy our implementation in the continental P4 Lab testbed and conduct the first hardware-based comparison of polynomial calculation, used in M-PolKA and PolKA, with traditional approaches: Table-Based L2 and LB-SR.

5.4.3.1 Setup description

RARE[15] (Router for Academia, Research, and Education) is an effort under the GÉANT 3rd program[16] focused on creating a routing software platform solution that can fit research and education use-case purposes. The project deployed a P4 Lab distributed among various European countries, as presented in Figure5.8, and is also expanding to Brazil (via RNP) and the US (via Starlight). The testbed used by our PoC comprises four Intel/Barefoot Tofino WEDGE100BF32X switches

[15] https://wiki.geant.org/display/RARE
[16] https://www.geant.org/Projects/GEANT_Project_GN4-3

powered by the TOFINO Network Processor Unit (NPU). They are geographically spread through the following locations: Amsterdam (AMS), Frankfurt (FRA), Poznan (POZ), and Budapest (BUD). As illustrated by Figure 5.8, the setup consists of two possible paths: i) the shortest (AMS-FRA); and ii) the longest (AMS-POZ-BUD-FRA). A 10Gbps optical link provides the inter-switches connectivity. Two edge nodes (H_1 and H_2) are in charge of generating traffic according to the packet size and forwarding method. To this end, we have used the *pktgen-dpdk*, version 19.12, with DPDK version 19.11.5, capable of sending custom packets via *pcap* files. Although DPDK is a fast-packet processing framework that allows *pktgen-dpdk* to generate 10 Gbit rate traffic with 64 bytes frames, the number of Packet Per Seconds (pps) is imposed by the network interface card (NIC) limitations.

5.4.3.2 CRC Polynomial $\mod$ Feature in ASIC Tofino

The ASIC Tofino Architecture (TNA) [P4 2018] supports the specification of an arbitrary CRC polynomial of 16 and 32 bits. The Code 5.4 shows how to initialize and calculate the hash value in the TNA, by using a set of parameters defined by its extern CRC polynomial $\mod$ in hardware. Note that it is possible to configure all the parameters of the CRC generator polynomial, as required by M-PolKA and PolKA to set the *nodeID*s in each switch.

```
/* ... */
CRCPolynomial<bit<16>>(
  coeff    = nodeID,
  reversed = false,
  msb      = false,
  extended = false,
  init     = 16w0x0000,
  xor      = 16w0x0000) poly;
Hash<bit<16>>(HashAlgorithm_t.CUSTOM, poly) hash_algo;
/* ... */
```

Code 5.4: CRC Polynomial $\mod$ in the TNA

5.4.3.3 Timestamp Header for Forwarding Latency Measurements

To measure latency into the P4 pipeline, Tofino's ASIC gives access to the following hardware timestamps, as illustrated in 5.9:

- ingress global timestamp: 48-bit field that represents the timestamp (ns) taken upon arrival at ingress control.
- egress global timestamp: 48-bit field that represents the timestamp (ns) taken upon arrival at egress control.

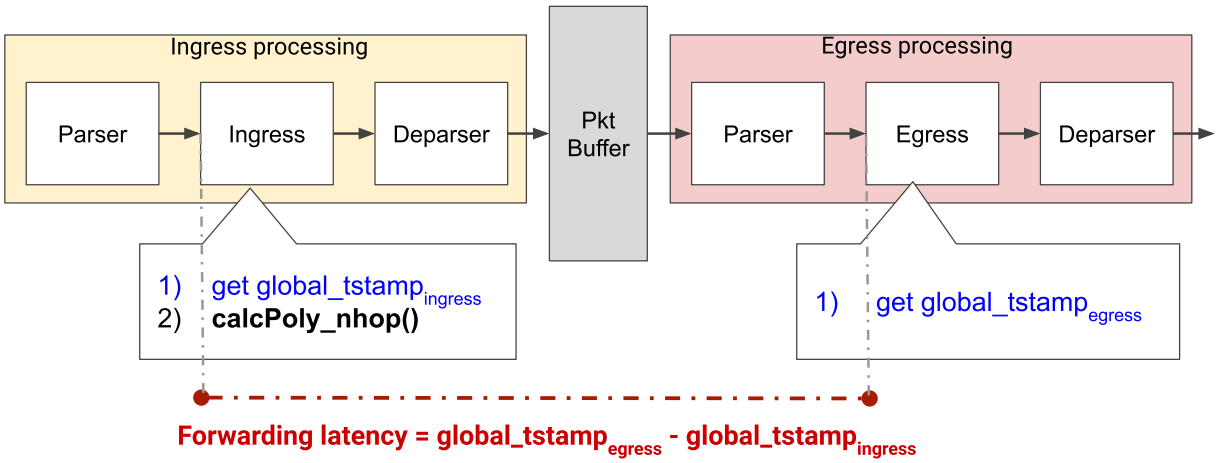


Figure 5.9: Tofino P4 Pipeline [Networks 2017].

As shown in Code 5.5 and Figure5.9, when a packet arrives at the data plane, we update the IP options by adding the global timestamp in the ingress control and, further, the egress control's global timestamp.

```
/* ... */
control Ingress(
  /* ... */
{
  apply {
    hdr.int_count.fim = ig_prsr_md.global_tstamp;
    /* ... */
  }
}
/* ... */
control Egress (
/* ... */
{
  apply {
    hdr.int_count.ini = eg_prsr_md.global_tstamp;
    /* ... */
  }
}
/* ... */
```

Code 5.5: Timestamps in the TNA

5.5 Performance Evaluation: from emulation to testbed deployment

This section evaluates M-PolKA and LB-SR for: (i) end-to-end tests in the emulated prototype, and (ii) single-hop latency test in the hardware prototype. Also, it

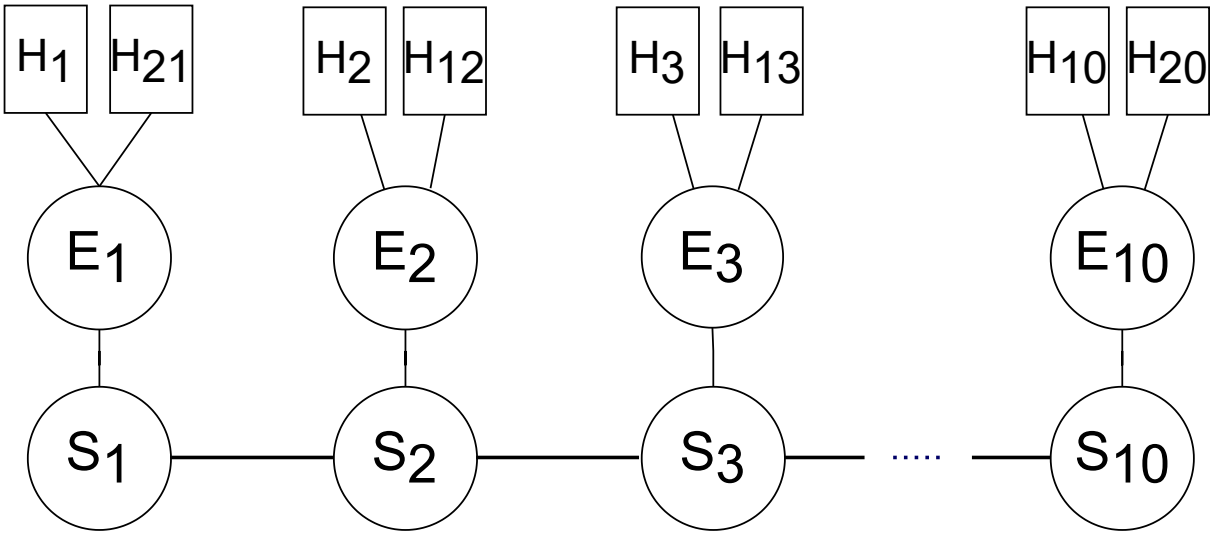


Figure 5.10: Linear topology.

demonstrates path migration, failure recovery, and multipath exploration in M-PolKA. We considered Ethernet frames of 98 Bytes as small packets and frames of 1242 Bytes as big packets for the tests. The average and standard deviation are presented in all the results.

5.5.1 E2E tests in emulated prototype

The test uses the linear topology of Figure 5.10 to compare M-PolKA and LB-SR as the number of hops increases in the core network (e.g., from 0 for path $H_1 \rightarrow \{H_{21}\}$ to 9 for path $H_1 \rightarrow \{H_{10}, clone(H_{20})\}$). It envisions to show how much the *routeID* size impacts the forwarding latency by increasing the number of hops. It is important to emphasize that we got all the measurements regarding the last cloned packet. Hence, the following experiments were executed:

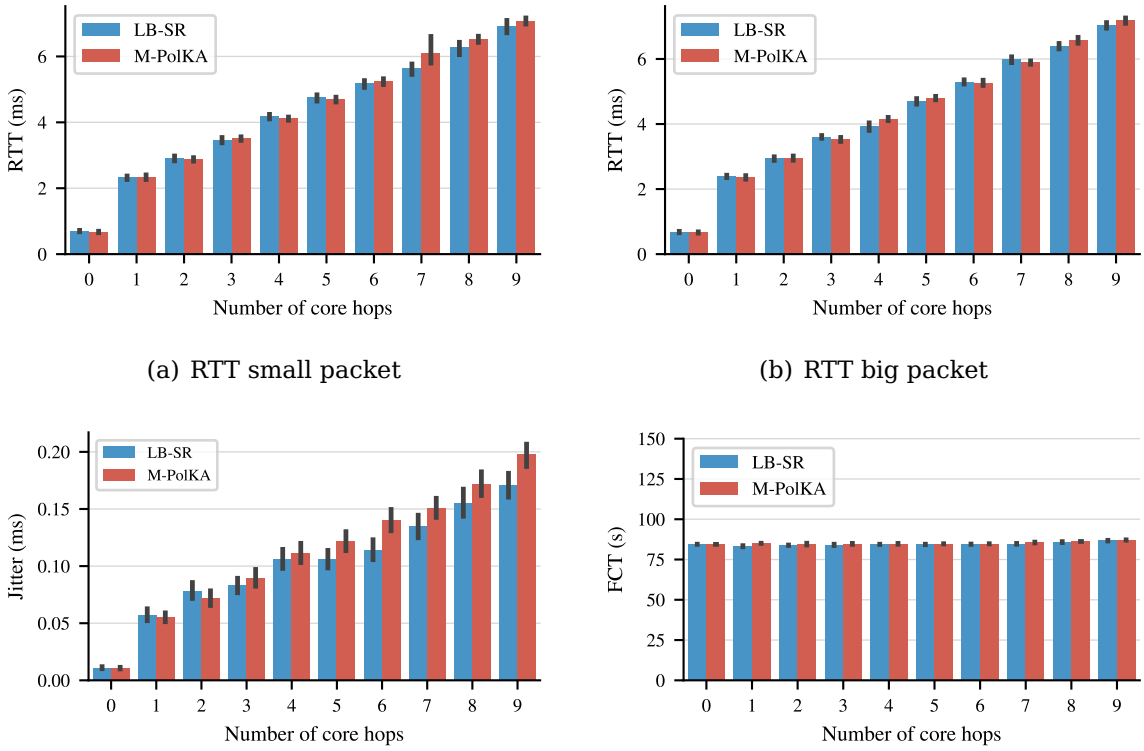
1. round trip time (RTT): host H_1 sends 1 ICMP packet/s during 60 s to each of the other hosts by using `ping` tool;

2. jitter: host H_1 sends a UDP traffic of 5 Mbps (half of the link capacity) with big packets to each of the other hosts during 60 s by using the `iperf` tool; and

3. flow completion time (FCT): host H_1 transmits a file of 100 Mb with big packets over a TCP connection to each of the other hosts by using the `iperf` tool (3 repetitions).

Figure 5.11 shows the comparison between M-PolKA and a simple List-based SR solution (Sourcey) for RTT, jitter, and FCT experiments.

In the RTT experiment results, shown in the Figure 5.11(a) and Figure 5.11(b), it is possible to observe that the RTT grows linearly with the increase of the number



(a) RTT small packet

(b) RTT big packet

(c) Jitter

(d) FCT

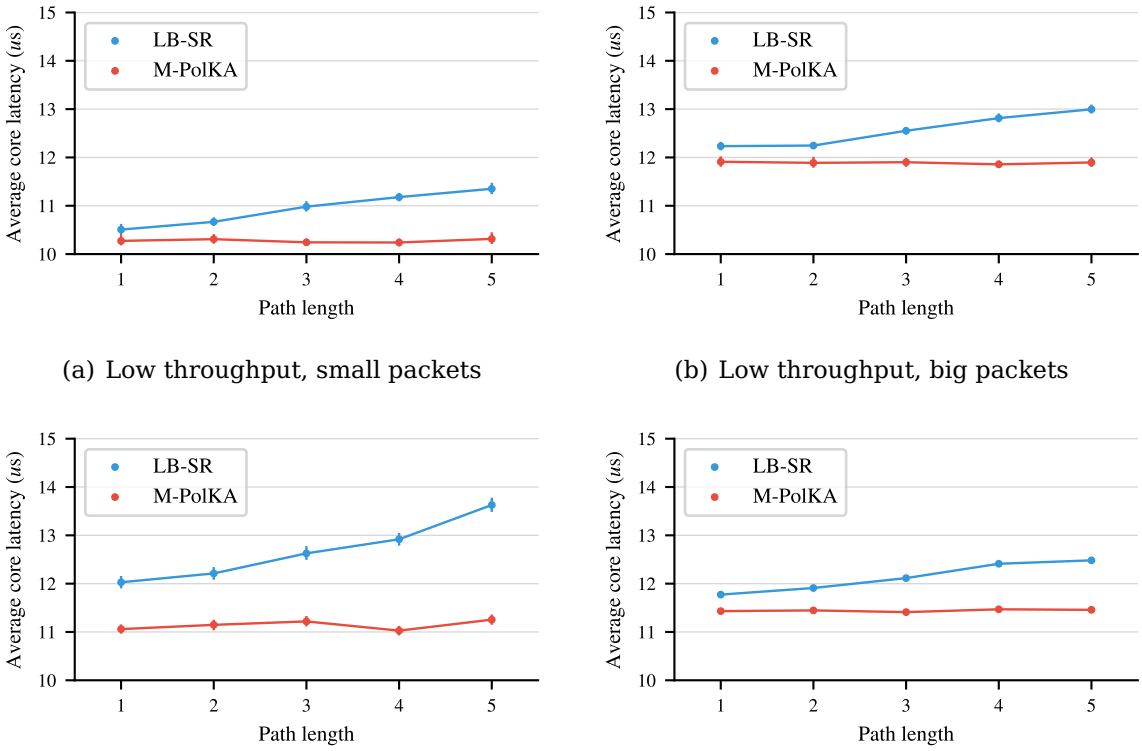
Figure 5.11: Linear scenario: comparison between LB-SR and M-PolKA.

of hops for both solutions. The LB-SR solution presents better RTT performance than M-PolKA, and for both solutions the standard deviation is small and in the same order of magnitude. Besides, there is no significant difference in the results for different packet sizes in the RTT experiments. This is because `Mininet-P4` does not consider transmission time in the emulation. In addition, jitter (Figure 5.11(c)) is small and equivalent for both solutions. Finally, the FCT experiment (Figure 5.11(d)) shows that both solutions require approximately the same time to transfer the file and the standard deviation is small.

The fact that LB-SR has a better RTT performance than M-PolKA is related to two facts: an LB-SR loses one SR header per hop, so the average packet header size is smaller than the fixed header used by M-PolKA, and the CRC operation for M-PolKA in this emulated prototype is executed in software. Nevertheless, the difference between the two solutions is small and can decrease if the CRC operation is performed in hardware, as shown in the Sections 5.5.2, 5.5.3, and 5.5.4.

5.5.2 Core latency evaluation in SmartNIC deployment

The goal in this experiment is to measure the core forwarding latency for a single hop in M-PolKA and an LB-SR when the path length increases. We consider the



(a) Low throughput, small packets

(b) Low throughput, big packets

(c) High throughput, small packets

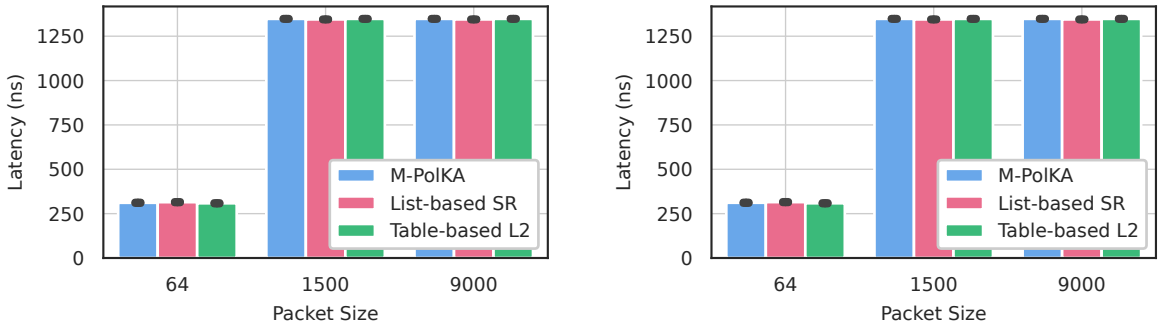
(d) High throughput, big packets

Figure 5.12: Comparison of LB-SR and M-PolKA test cases.

path length as the number of core nodes included in the SR header to reach the destination. For each test execution, the traffic generator tool at TX varies the IP destination address. The last digit of the IP destination address represents the number of core nodes (e.g., if the IP destination is 10.0.100.1, the number of hops to the destination is 1, while IP destination 10.0.100.5 represents 5 hops to the destination). Depending on the number of hops, the edge encapsulates the appropriate SR headers (e.g., 5 hops, 5 SR headers in an LB-SR). The following experiments were executed for small and big packets: (i) low throughput: one ICMP pps, 100 packets in total, generated with `ping` tool; and (ii) high throughput: 1Gbps UDP packets, 1000 packets in total, generated with `pktgen` tool.

Figure 5.12 compares the test cases for LB-SR and M-PolKA. In each test, the average latency and standard deviation in M-PolKA are small when the path length increases, while in an LB-SR, the average latency grows linear when the path length increases. This linear increase in latency measurements for an LB-SR is emphasized in the test case with high pps values (Figure 5.12(c)) when the standard deviation is a bit higher for both M-PolKA and LB-SR due to the stress in edge and core elements. More investigation needs to be carried out in a hardware prototype that allows multi-hop tests, such as switches with Intel Tofino Architecture[17]. Despite

[17]https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-



(a) 1Gbps (b) 10Gbps

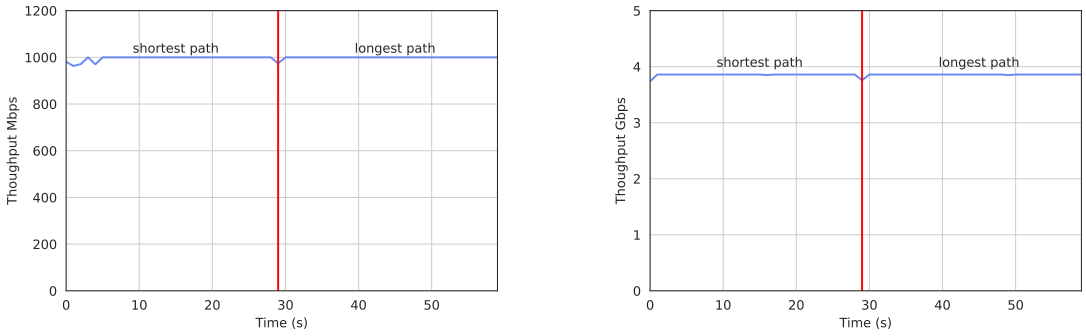
Figure 5.13: Forwarding latency in M-PolKA vs. List-based SR vs. Table-based L2.

this, the results collected so far indicate that M-PolKA implementation by using CRC hardware is promising and can offer equivalent RTT and jitter performance to an LB-SR approach.

5.5.3 Core latency evaluation in P4 Tofino-switches testbed

The goal is to compare the core forwarding latency for the three methods by analyzing different packet sizes in the throughput rate of 10Gbps. To this end, we measure the forwarding latency at BUD during the transmission from H_1 to H_2. When a packet arrives at BUD, the P4 application performs a sequence of match and action operations at the Ingress control (polynomial calculation for M-PolKA), as usual; however, it also adds in the IPv4 options field two 48-bit timestamps regarding upon arrival at the ingress and egress control blocks, as shown by Figure 5.9. Therefore, we have obtained the forwarding latency measurements about the whole packet's trajectory into four stages: ingress control, ingress de-parser, packet buffer, and egress parser. Finally, all packets are captured and parsed offline to extract the forwarding latency for each packet by subtracting the *egress global timestamp* from the *ingress global timestamp*. Our experiments were performed by using the *pktgen-dpdk* tool to generate traffic for small (64 bytes), medium (1500 bytes), and jumbo frame packets (9000 bytes) with a throughput of 10Gbps of UDP packets. In total, we captured 10000 packets as samples to obtain the average and the standard deviation about each forwarding method. As shown in Figure 5.13, the average for small packets is on the scale of about 300ns, whereas for medium and jumbo frames, the average is about 1300ns. The results show the capability of M-PolKA to run in high-performance switches and the potential of either PolKA or M-PolKA to replace well-known forwarding methods, such as table-based, with the same performace.

switch.html



(a) 1Gbps (b) 4Gbps

Figure 5.14: Migration from short path to long path.

5.5.4 M-PolKA fast-path reconfiguration in RARE testbed

The first experiment starts a 1Gbps UDP traffic (Figure 5.14(a)) from H_1 to H_2. In the second experiment, we start a 4Gbps UDP traffic (Figure 5.14(b)) from H_1 to H_2. The throughput measurements were collected by using the *bwm-ng* tool at H_2. In both experiments, we start using the blue path (short) and, as shown in Figure 5.8, at 29s, the red path (long) is selected by the traffic engineering. To perform the path reconfiguration, the SDN Controller is in charge of modifying a single flow entry at the edge node H_1 to achieve the destination H_2. The only field that has to be modified is the *routeID* to embed the new route through the red path ($H_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow H_2$). Once this single operation is executed, all the packets of flow into the blue path will be tagged with the new *routeID* and steered to the red path. Therefore, it is important to note that a small packet loss is detected at the 30s in the destination, demonstrating the fast-reconfiguration capability of M-PolKA.

5.6 Use case: Multipath routing for reliable communication

5.6.1 Use case architecture design

The design principles of this use case are motivated by the high-reliability requirements for the new-generation networks. For our use case architecture, we have separated it in four layers, as shown in Figure 5.15:

(i) the physical infrastructure layer;

(ii) virtualization abstraction;

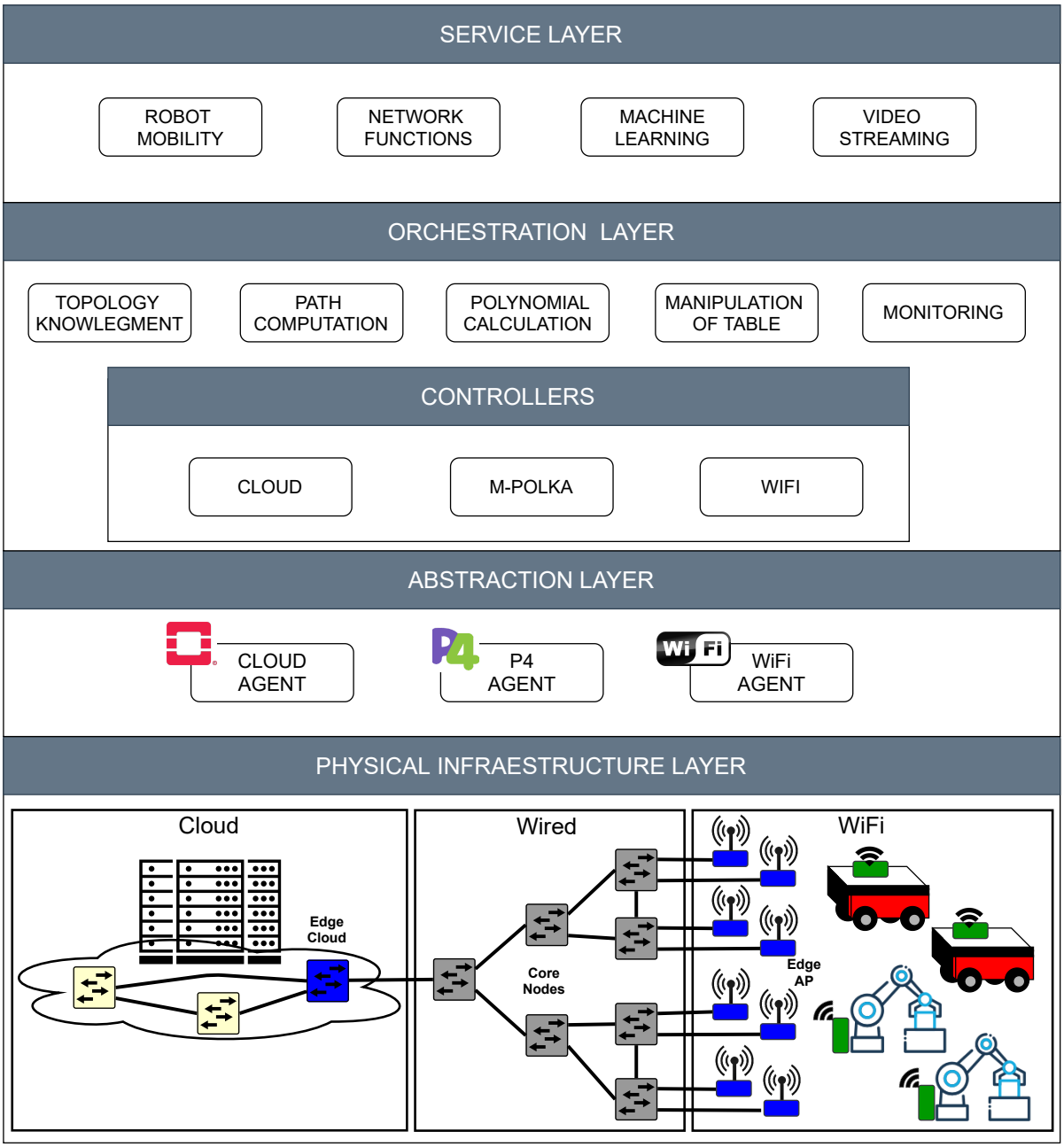


Figure 5.15: Use case Orchestration Framework Architecture.

(iii) the control and orchestration layer;

(iv) and the service layer.

The separation of functionalities and components among the layers gives us a modular architecture and makes its implementation and third-part integration easier. Therefore, we design and deploy this use case envisioning the orchestration layer with components (topology, path computation, polynomial calculation) that unified the orchestration software, but delegating the controller per technology with heterogeneous resources (cloud, network and WIFI).

5.6.1.1 Physical infrastructure layer

To meet reliability requirements, the physical infrastructure of the M-PolKA architecture assumes the existence of a physical topology with sufficient path redundancy to allow the implementation of mechanisms to guarantee a reliable communication. It is worth mentioning that the multipath computation algorithms to find the best routes with node-disjoint, link-disjoint or even non-disjoint paths to improve fault-tolerance will be future work of this thesis.

Taking advantage of the network's physical redundancy, M-PolKA can implement multipath forwarding to support multicast communication, enabling better network resource utilization based on the fast reaction of network changes. The physical infrastructure is composed of four specific elements:

(i) **Edge cloud nodes** that add and remove the *routeID*s from the packets coming and arriving from the cloud infrastructure; The traffic splitting at the edge for load balancing over multiple paths has not been fully explored in this work.

(ii) **Edge AP nodes** perform polynomial operations for the forwarding process to know which WiFi ports belong to the *routeID* in which they have to steer. They also add and remove the *routeID*s from the packets when receiving and sending to the STA nodes, respectively;

(iii) **STA nodes** analyze the traffic type and perform the function chaining (SFC) accordingly (i.e., duplicated packet or not). The nodes also provide a programmable data plane to steer the traffic to the Edge AP Nodes according to given traffic engineering;

(iv) **Core nodes** are in charge of performing polynomial operations for the forwarding process to steer the traffic to one or any ports.

Implementing all these operations in P4 improves latency once operations and decisions can be made directly at the hardware level. Furthermore, P4 is a platform- and protocol-independent architecture that takes full advantage of writing our own forwarding data plane (i.e., the same P4 code for virtual and bare-metal switches).

5.6.1.2 Virtualization Abstraction

In the Virtualization Abstraction, each agent creates a unified SB API to the controllers to abstract the interface with the hardware independently of the vendor, version, and model. When the agent receives a request from a control plane service, a control plane core service uses an SBI to communicate. Then, the agent uses drivers to interact with underlying network devices and then uses the information to

reply to the request. In each driver, we translate requisitions from the SBI to instructions that a specific device can interpret. We have implemented three Virtualization Abstraction agents, as follows:

1. **Cloud Agent** is in charge of interacting with IaaS or PaaS solutions, such as OpenStack, Kubernetes, or COPA [Both et al. 2019]. Indeed, the agent is used to create a common interface that interacts with the Iaas and/or PaaS architectures to form end-to-end connectivity across different domains.

2. **P4 Agent** is in charge of adding, removing, and updating rules in the tables in a P4 switch. It can also set irreducible polynomials in edge AP or edge core nodes and cloning sessions for all types of nodes, as shown in Section 5.3.3. The P4 Agent uses a driver to translate the instructions to a P4 Runtime in the BMv2 or a P4 Runtime ASIC Tofino. For the ASIC Tofino, we have developed the driver using a proprietary Python Library that interacts with the switch device.

3. **WiFi Agent** sets the WiFi parameter (e.g., channel, SSID, BSSID, and power.) from messages delivered by the WiFi controller and sends information about the SNR, association, and disassociation of each STA. Also, the agent can take some forwarding decisions based on its scope of visibility, such as link failure or low SNR.

5.6.1.3 Control and orchestration layer

The design of new-generation networks requires control and orchestration facilities to offer a harmonious operation of all the architectural elements. This layer represents the heart of the architecture, whose main function is to control and manage the multiple domains through an integrated manipulation of the underlying physical layer elements. By operating under the principles of software-defined networks and microservices, the controllers in this network can be found in distributed schemes. Each element can take better forwarding decisions, ordered via a central entity or local. As an outcome, we have a faster manipulation time with the data plane since the devices can directly do certain triggers instead of going to the upper layers. It alleviates the control and orchestration layers and, thus, provides a more scalable and reliable architecture.

Distributed throughout the switching elements, software agents monitor all the nodes and their links and send information from all domains (i.e., wired and wireless) to the orchestrator by using a common AMQP interface. With this information, the orchestrator can maintain updated information about the network's active topology and use the available network diversity accordingly (**Topology Knowlegment** and

Monitoring). The best routes are calculated for simple communications between source and destination (**Path Computation**). Depending on the network conditions and the end-to-end requirements, the orchestrator can explore multiple paths to ensure more communication reliability. Besides, the orchestrator can include backup routes, increasing the resilience of the *routeID* in case of failure.

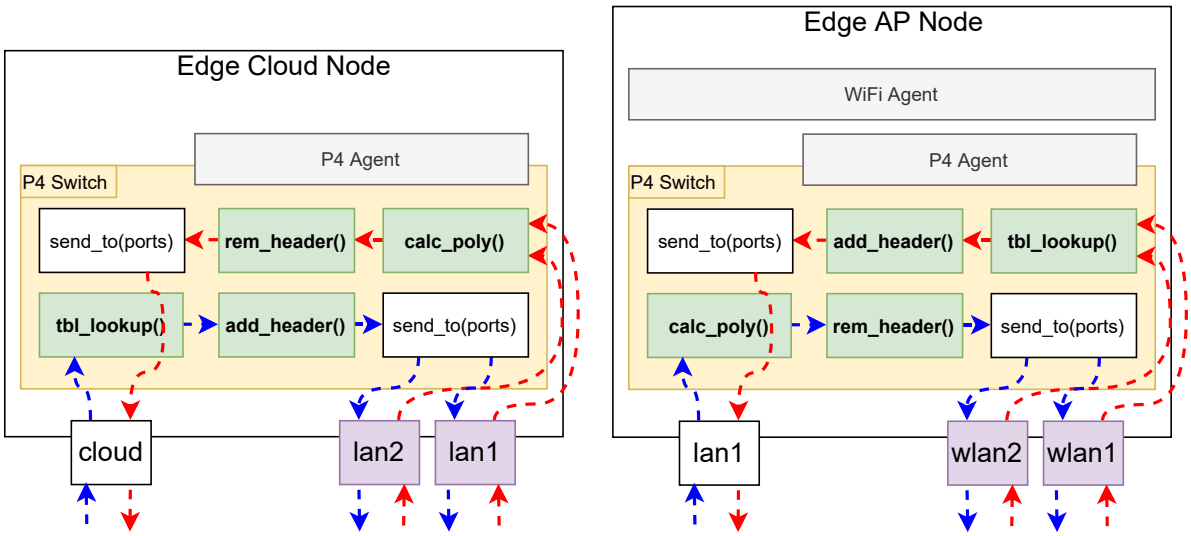
With the available routes calculated, the control and orchestration layer has on its hand a multicast RNS SR scheme to ship routing information being transported by the packets themselves. Based on the nodes to be traversed and the transmission state at each node, **Polynomial Calculation** module generates a *routeID* for an entire end-to-end communication (simple route or multipath route) across different domains. Then, the *routeID* is added as a new packet's header from the *Edge Cloud* and *Edge AP* nodes via **Manipulation of Tables**. As already mentioned, the edge nodes are the elements in charge of adding the M-PolKA header in each packet that enters the M-PolKA domain. The orchestrator, via the controllers, adds, removes, or updates the data plane using the P4 agent. The P4 data plane's implementation to compute the irreducible polynomial used in our SR scheme and the polynomial operation were discussed more deeply in the section 5.3.

5.6.1.4 Service layer

This layer comprises all user applications and other tools used in the architecture implementation and validation stages. Those applications can push ahead an additional integrated layer with the control and orchestration layer in order to deliver, for example, applications for handover focused on robot mobility and traffic engineering by using machine learning techniques. This layer also embraces tools in charge of generation traffic (e.g., iperf and pktgen) and video streaming to be used for architecture validation.

5.6.2 Implementation Design

In our architecture, there are nodes responsible for inter-connect traffic from different domains that do not understand M-PolKA. For instance, for incoming traffic, there is a need to add in the packet a M-PolKA header with the appropriate *routeID*. Nevertheless, for outgoing traffic, there is a need to remove and deliver according to the transmission state obtained by the operation mod between *nodeID* and *routeID*. Hence, this section aims to explain how we implement the elements edge and STA nodes using P4 into wired and wireless domains. For this purpose, we have divided the following sections as (i) edge cloud nodes (5.6.2.1); (ii) edge AP nodes (5.6.2.2); (iii) STA nodes (5.6.2.3).



(a) Edge Cloud (b) Edge AP

Figure 5.16: Edge Nodes Architecture.

5.6.2.1 Edge Cloud Nodes

As shown by Figure 5.16(a), during the incoming traffic from the cloud domain, the edge cloud node initially executes a lookup that returns the *routeID* and the transmission state associated with the flow. After that, the edge cloud node adds the M-PolKA header with the associated routeID and steers the traffic to one or more ports on that hop. Incoming traffic from M-PolKA domain to the cloud domain, the edge cloud node calculates the polynomial, removes the M-PolKA header, and last steers the traffic based on the polynomial operation result.

5.6.2.2 Edge AP Nodes

Similar to the edge cloud nodes, the edge AP node adds and removes the M-PolKA header based on a lookup table operation, which results in the *routeID* and transmission state, as shown by Figure 5.16(b). Furthermore, the edge AP node has a WiFi agent integrated into the control loop systems that can act to some WiFi event (e.g., migration, link failure). As well as happens in the cloud domains, the WiFi domains do not understand M-PolKA. That's why we have to perform the add and remove the M-PolKA's header for each packet.

```
action hash_() {
    hash<bit<32>, bit<16>, tuple<bit<16>,bit<16>,bit<8>,bit<8>,bit<32>,
    bit<32>,bit<16>,bit<16>,bit<16>,bit<16>>
        (meta.fw_index,
        HashAlgorithm.crc32, 16w0,
        {   hdr.ipv4.totalLen,
            hdr.ipv4.identification,
```

```
            hdr.ipv4.ttl,
            hdr.ipv4.protocol,
            hdr.ipv4.srcAddr,
            hdr.ipv4.dstAddr,
            hdr.ipv4.hdrChecksum,
            meta.srcPort,
            meta.dstPort,
            meta.t_checksum },
        REGISTER_LENGTH); // REGISTER_LENGTH is the list size
}
action read(bit<32> hash) {
    has_pkt.read(meta.checksum, hash);
}
action write(bit<32> hash) {
    has_pkt.write(hash, meta.t_checksum);
}
/* ... */
apply {
        if (hdr.ipv4.isValid()) {
            do_hash();
            get_hash(meta.fw_index); // check duplicated packets is in the list
            if (meta.has_pkt == 0) {  // is a new packet
                set_hash(meta.fw_index); // add hash to the list
            } else {
                if ( hdr.tcp.isValid())
                    if (meta.checksum == hdr.tcp.checksum)
                        drop(); // drop duplicated packet
                if ( hdr.udp.isValid() )
                    if (meta.checksum == hdr.udp.checksum)
                        drop(); // drop duplicated packet
            }
            /* continues performing M-PolKA */
            do_mpolka();
        } /* ... */ }
```

Code 5.6: STA Node P4 algorithm

5.6.2.3 STA Nodes

One of our STA nodes' main capabilities is to perform an SFC to filter duplicated packets before sending them to the application. In our new P4-based implementation, instead of offloading the traffic to traditional VNF on vCPU, we decompose and deploy this functionality into small embedded Network Functions (eNFs), as previously proposed by the work [Mafioletti et al. 2020]. Our hardware-agnostic packet duplication scheme is described in Section 4.2.2.4; the VNF in the chaining is responsible for the flow de-duplication function.

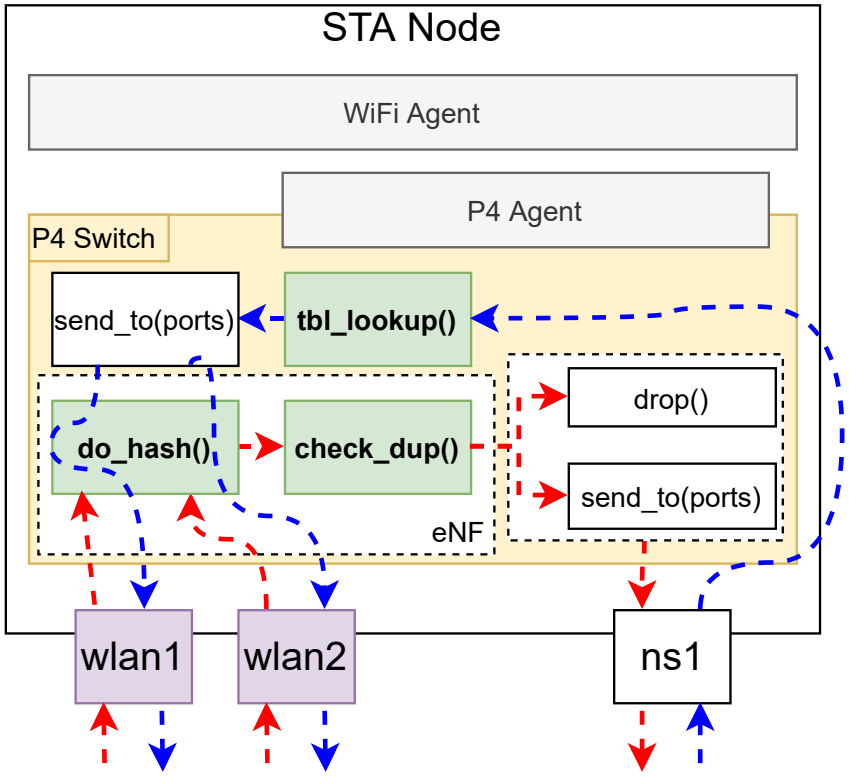


Figure 5.17: STA Architecture.

Instead, as shown by Figure 5.17 and Code 5.6, the eNF receives a packet denoted by p_i, generates a hash $do\_hash(p_i)$ using the TCP/IP headers, and compares if the current hashed packet $check\_dup(p_i)$ already exists in a circular hash list denoted by β. If so, then the packet p_i is automatically dropped, otherwise, $check\_dup(p_i) \notin \beta$, then the packet is forwarded to its namespace, and then $do\_hash(p_i)$ is added to β. For this matter, we save end-host server CPU cores and avoid overheads generated by the whole virtualization stack while achieving lower latency for service requests.

5.6.3 Proof-of-concept and experimental validation

To evaluate the main functionalities of M-PolKA compared to the LB-SR, an emulated prototype was implemented to evaluate end-to-end scenarios, where we increase the communication reliability by exploiting path diversity. As an outcome, we demonstrate (i) agile path migration during the handover process in the next generation networks (Section 5.6.5.1); (ii) packet duplication to provide redundant path during an end-to-end communication in a given time (Section 5.6.5.2); and (iii) fast failure recovery to recover the communication during some failure event (Section 5.6.5.3).

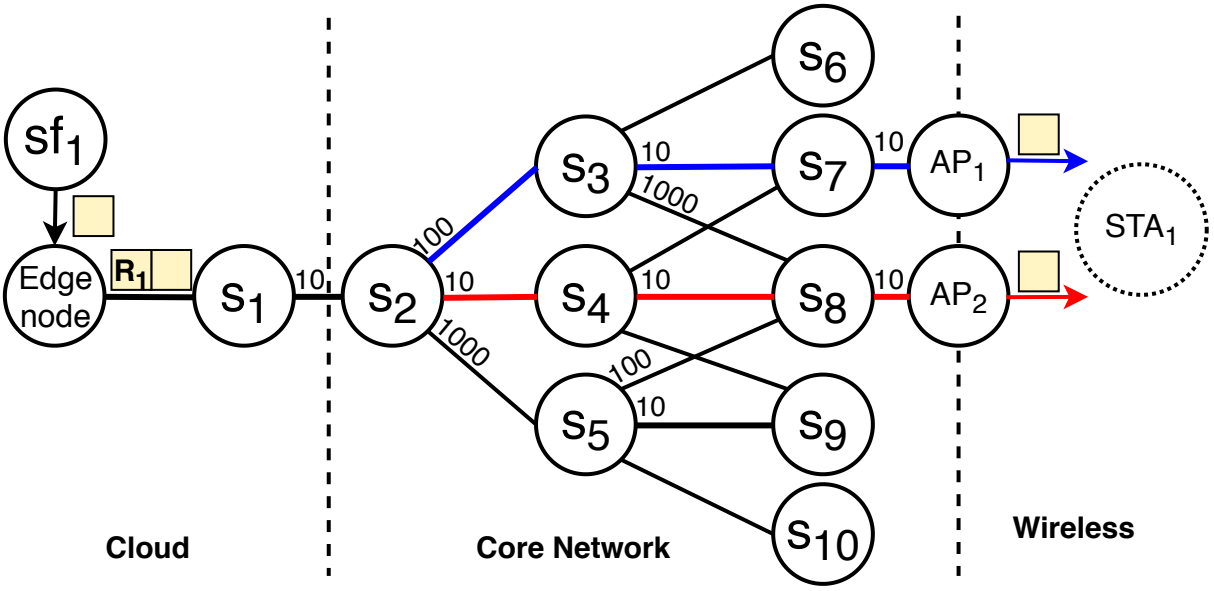


Figure 5.18: Multipath routing for reliable communication use case

5.6.4 Agile and expressive orchestration deployment

5.6.4.1 P4 architecture and target

The software switch `bmv2 simple_switch` with the `v1model` architecture was selected as the target for this prototype, once it supports all the functionalities required by M-PolKA, such as the configuration of CRC polynomials. Also, we use the `clone3` primitive action (P4_16) to enable the multipath feature. The number of clone session IDs is limited to 65535 in the `bmv2 simple_switch`. It is important to highlight that this software switch is a user-space implementation focusing on feature testing. There are other high performance implementations of P4 software switches and compilers (e.g., PISCES[18], P4ELTE[19], and MACSAD[20]), but they do not yet cover all the features required by M-PolKA. As these implementations arise, it will be possible to test our prototype with higher loads. For the time being, the solution was to limit the link rates to 10Mbps in our emulated prototype to avoid reaching the processing capacity limits of `bmv2 simple_switch`.

5.6.4.2 Setup description

The setup consists of one server Dell PowerEdge T430, with one Intel Xeon E5-2620 v3 2.40 GHz processor and 64 GB of RAM. To build our emulated environment, we used Mininet-WiFi[Fontes et al. 2015] with P4[21], which becomes a tool that augments the well-known Mininet emulator by including virtual wireless stations

[18] http://pisces.cs.princeton.edu/
[19] http://p4.elte.hu/
[20] https://github.com/intrig-unicamp/macsad/
[21] https://github.com/jafingerhut/p4-guide

(STA) and access points (AP), and, likewise, allows running P4 programs on different switches. This functionality is crucial to emulate wireless and wired networks, which require different P4 programs for the edge AP, edge cloud, core, and STA elements.

5.6.4.3 Control plane implementation

It has two main functionalities: (i) for core: compute *nodeID*s, and configure core switches with their respective identifiers; and (ii) for the edge: compute routing paths for the traffic flows, calculate *routeID*s for these paths, and configure table entries in edge switches that will be responsible for encapsulating *routeID*s. For the RNS computation of *nodeID*s and *routeID*s, as described in Section **??**, we developed a Python library[22] and application that uses the package `galoistools` of the `sympy` library[23] for GF(2) arithmetic operations. For programming each core and edge switch, we developed a control plane application in Python that communicates with the switches using the CLI commands provided by the `bmv2 simple_switch`. The control messages format is defined by an API that connects to a Thrift RPC server running in each switching process.

5.6.4.4 Header size

The `bmv2 simple_switch` only supports the specification of CRC polynomials of 16 and 32 bits. Therefore, although our tests could use much smaller degrees, our PoC must adopt polynomials of degree 16. As the diameters of our test topologies are smaller than 10, the size of the M-PolKA header was defined as 160 bits. To a fairness comparison, we defined the LB-SR header as 16 bits (bos and port); therefore, for 10 hops, the array of headers has 160 bits.

5.6.5 Exploitation of path diversity for reliability

This section explores a software-defined wireless network as a use case to demonstrate how M-PolKA can exploit path diversity for enabling reliable communication. To this end, three experiments were designed as a proof-of-concept of the following reliability functionalities: agile path migration, packet duplication, and fast failure reaction.

Figure 5.18 shows the experiment scenario with three network domains: cloud, core network, and wireless. The cloud composes a virtualized environment responsible for providing service function (SF) to a specific mobile device group. In the core network, we have 9 switches (S_n) running core functionalities and two access points

[22] https://pypi.org/project/polka-routing/
[23] https://www.sympy.org/

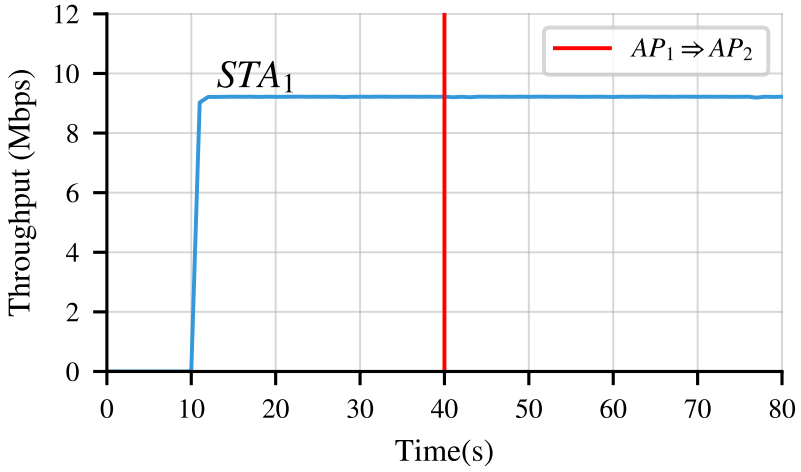


Figure 5.19: Agile path migration.

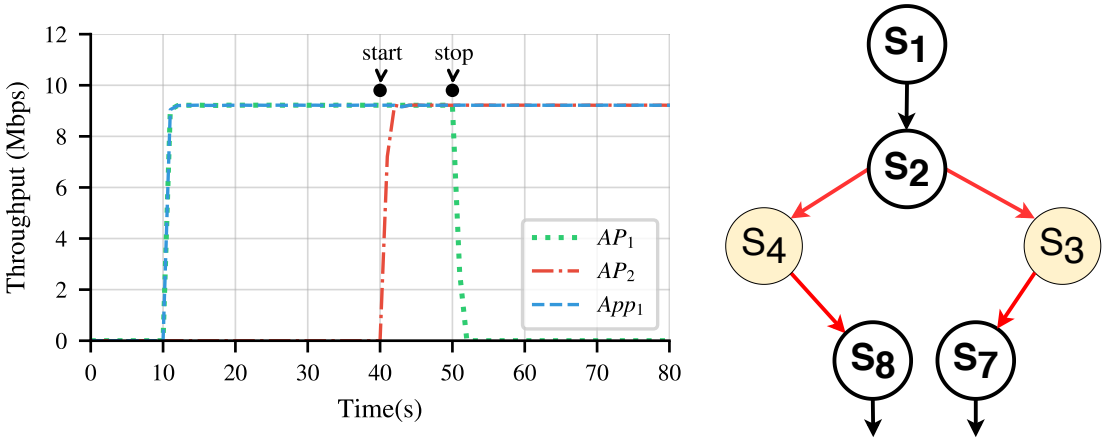
(APs) (AP_i) running edge functionalities. The wireless environment includes mobile elements (station - STA_i), where each STA has two wireless interfaces in order to provide multi-connectivity from different APs.

5.6.5.1 Agile path migration

This experiment shows how the traffic can be steered using SR for mobile elements with an agile path migration in the scenario of Figure 5.18. This feature can be used for improving reliability when the orchestrator identifies that a path migration can provide better perfomance.

As shown in Fig. 5.19, at 10 s, a flow ($SF_1 \rightarrow STA_1$) starts TCP traffics with big packets using the `iperf` tool. Initially, the flow is allocated to blue path ($S_1 - S_2 - S_3 - S_7 - AP_1$). The STA_1 is already connected to both AP_1 and AP_2. At 40s, the orchestrator starts triggering the handover process from the STA_1, once the AP_2 becomes the best option (e.g., signal-to-noise ratio). Hence, the orchestrator migrates the flow from the blue path to the red path ($S_1 - S_2 - S_4 - S_8 - AP_2$). Even though the STA_1 can be served by multiple APs, from this moment, the traffic now is only steered via AP_2.

To perform path migration, the SDN Controller only has to modify a single flow entry at the edge switch `Edge Node` to set a *routeID* to the destination STA_1. It modifies the *routeID* to steer the traffic through the red path. Then, once this single operation is executed, all the packets of flow that leave SF_1 are tagged with the *routeID* of the new path, and no packet loss is detected at the destination. Although this scenario emulates a simple example of horizontal handover, the same method can be used for vertical handover [Kassar et al. 2008] solutions in heterogeneous networks.



(a) Throughput at AP_1, AP_2 and STA_1 (b) Multipath

Figure 5.20: Duplication of traffic from S_2

5.6.5.2 Packet duplication

For multipath scheme, our architecture implements a VNF-based solution proposed by [Guimaraes S et al. 2020]. In this work, once the data plane in the $STAs$ identifies the type of flow, it determines how the packet will be forwarded: i) for packet duplication, the data plane sends the traffic to be treated by the VNF at the STA before sending to App_1; ii) for the normal packet, the data plane redirects the traffic strictly to the destination, the VNF App_1. However, during the whole experiment, the traffic is forwarded strictly to the VNF before being sent to the App_1 by the STA, even in simple path conditions.

Figure 5.18 shows an example scenario to provide reliable communication between $SF1$ and STA_1. The list of switches for the segment ($SF1 \rightarrow STA_1$) that is used for providing the multipath scheme is: nodes S = $\{S_1, S_2, S_3, S_4, S_7, S_8\}$. A *routeID* carries the set of transmission state of each node, where it steers between simple and multipath traffics.

Initially, the path is represented by the nodes S = $\{S_1, S_2, S_3, S_7\}$, which achieves the STA_1 using AP_1. During the handover process, before stabilizing the connection with the AP_2, the control plane sets the transmission state on node S_2 to 110, which means to steer the packets to the nodes S_3 and S_4, ports 2 and 3, respectively. Thus, when the link becomes stable between STA_1 and AP_2, the control plane sets the transmission state on node S_2 to 10, which steers the traffic to only S_4. It implies modifying the *routeID* during the element's mobility as the path has to contain a new set of transmission state of the nodes.

To integrate with the wireless mechanism in our prototype, we developed an orchestrator that modifies the traffic according to the elements' mobility (STA). It is not the focus of this work to identify the optimal time based on, for example,

`signal-to-noise ratio` to do the transitions between different access points. Additionally, we modified the core application to perform the packet cloning forwarding based on the $\mod$ operation between nodeID and *routeID*.

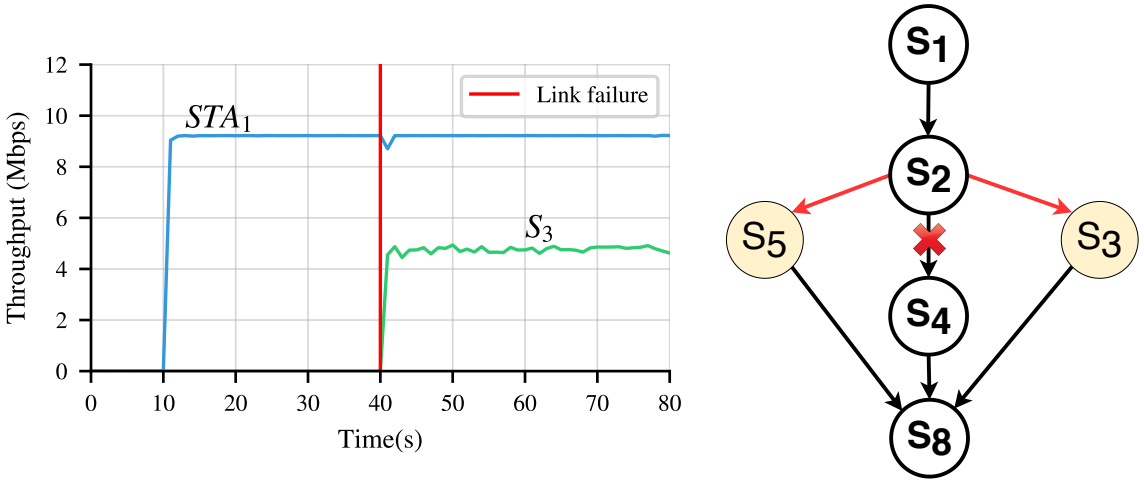
Figure 5.20 shows throughput measurements using the `bwm-ng` tool at STA_1, AP_1, AP_2 (Figure 5.20(a)), the traffic is duplicated through S_3 and S_4 during the handover process. At 10 s, we start a 10 Mbps UDP traffic from SF_1 to STA_1 passing through AP_1. At 40 s, the transmission state on node S_2 is initiated to pass through $S_3 - S_7 - AP_1$ and $S_4 - S_8 - AP_2$. At App_1 in STA_1, the traffic constantly perceives about 10 Mbps as the VNF discards duplicated packets before sending to the STA_1. There is no throughput degradation during migration between APs. Therefore, our scheme could increase reliability once there is a high probability of losses during the handover processes (mobility interruption time - MIT). We do not need to maintain the state in the whole path with our approach as all transmission states' nodes were already included in the *routeID*.

5.6.5.3 Fast failure recovery

This experiment aims to show an example of how M-PolKA can take advantage of special RNS properties. More specifically, it explores a property that states that the nodes' order in the path is irrelevant. Based on this property, we integrate a fast failure reaction mechanism proposed by KAR [Gomes et al. 2016] to the mobility scenario proposed in [Guimaraes S et al. 2020]. KAR proposes the concept of a resilient forwarding path, called protection path. The main idea is to proactively add redundant nodes in the *routeID* that are not part of the original route. When there is a link failure, packets are deviated from faulty links with routing deflections and may occasionally reach these redundant nodes responsible for guiding the packets back to the original route. In this way, there is no need to communicate with a controller (or even the source) to compute an alternative path because, as soon as the core node detects a failure, it randomly deflects packets to one of its healthy links.

Figure 5.18 shows an example scenario for fast failure properties ($SF1 \rightarrow STA_1$). The path in the core switches for the segment $SF1 \rightarrow STA_1$ is: nodes S = $\{S_1, S_2, S_4, S_8\}$ and ports O = $\{10, 10, 10, 10\}$. You can have paths called unprotected paths, because they do not add any redundant node for failure protection. Therefore, if any link of these paths fails, the packets will be dropped.

By applying the protection mechanism to generate the *routeID* of the segment ($SF_1 \rightarrow STA_1$), we add the extra nodes S_3, and S_5 with ports 1000, and 100, respectively. Thus, when link S_2-S_4 fails, S_2 deflects packets to any of its other links, and packets will be driven back to S_8, as shown in Fig. 5.21(b). Thus, the protected path is represented by nodes S = $\{S_1, S_2, S_3, S_4, S_5, S_8\}$ and ports O = $\{10, 10, 1000, 10, 100, 10\}$.



(a) Throughput at destination and S_3 (b) Protected Path

Figure 5.21: Fast failure reaction for failure of link S_4-S_6.

As the protected path already contains redundant nodes, no change in the *routeID* is needed when the failure happens.

To integrate KAR [Gomes et al. 2016] mechanism in our prototype, we developed a simple control plane application that causes link failures and makes port failure information available to the data plane by populating a table of faulty ports. Additionally, we modified the core pipeline to perform a lookup in this table before sending the packet to the output port.

If there is a hit, the packet is randomly deflected to one of the other healthy ports. Otherwise, the packet is transmitted normally. The generation of a random value within an interval is provided by `v1model` and could also be replaced by a hash function if the objective is always to select the same port per flow. Failure detection mechanisms are not in the scope of this work.

Figure 5.21 shows throughput measurements using the `bwm-ng` tool at STA_1 and S_3 (Fig. 5.21(a)). Results for S_5 were omitted, because they are similar to Fig. 5.21(a) as the traffic was uniformly deflected through S_3, and S_5 after the failure. At 10 s, we start a 10 Mbps UDP traffic from SF_1 to STA_1. At 40 s, the link S_2-S_4 is disconnected.

At STA_1, the traffic perceives a small loss until the failure is signalized by the control plane and deflections start. Therefore, our scheme could react to failures without any packet modification as the redundant nodes were already included in the *routeID*.

5.7 Chapter Remarks

Herein, a binary polynomial representation of a fully stateless multipath SR approach, called M-PolKA, was proposed, implemented, and evaluated. M-PolKA extends and

generalizes PolKA by giving more expressiveness and providing multicast/multi-path functionality, opening the possibility to enable reliability functionalities in the new generation networks for any topology. Moreover, our P4-based emulated and hardware prototypes demonstrated that it is feasible to deploy RNS-based SR in high-performance network equipment by reusing CRC hardware, with performance equivalent to traditional routing approaches. As future work, this achievement has the potential to enable a new range of complex network applications that explore RNS intrinsic features, such as route authenticity, hybrid forwarding methods to save TCAM, and cross-layer programmability and slicing.

Chapter 6

Conclusion and Future Works

In this concluding chapter, we summarize this thesis and the work hereby presented to discuss the implications of our research. We then give our view concerning possible research directions that may emerge from our work and how other techniques can extend the network programmability for expressive and agile orchestration. Finally, we finish the thesis with our concluding remarks.

6.1 Thesis Summary

The broader considerations of this thesis only grasp the tip of the iceberg that seeks to challenge the commonly held view that the network needs to provide cross-layer programmability as an enabler for orchestration across heterogeneous resources. Given that, we have explored various aspects of expressiveness and agility during the orchestration process in which led us to tackle the following questions: "how to extend SDN paradigm to set up physical layer parameters by a cross-layer programmability?"; "and, in parallel, how to provide more agility in the orchestration process in order to set up an E2E communication?".

The work presented in Chapter 3 and 4 explores these questions experimentally. The merit of our approach is on extending the range and the way to control physical layer (e.g., wireless and optical channels with adaptive bandwidth), link layer (e.g. new tableless forwarding mechanism;) so that these configurable parameters are exposed to create a cross-layer programmability abstraction. This abstraction is under the command of an orchestration process that provides functionalities (e.g. automatic scaling policies) so that it may set parameters across heterogeneous resources.

The orchestration paradigm designed in FUTEBOL's CF [Both et al. 2019] was extended by adding expressiveness and agility dimensions to its orchestration. An expressive orchestration is achieved by exposing physical parameters crossing the

network layers in order to boost the network programmability activated by a high-level API. Orchestration agility demands extremely fast route configuration involved in a datapath configuration. Orchestration agility is also required to meet quick handovers for seamless mobility support. These two new dimensions were pushed by applications with stringent requirements that involve providing ultra-reliable and low-latency communications with fast reconfigurability, such as cloud robotics [do Carmo et al. 2019a]. For this matter, we developed an architecture for better mobility management in WiFi networks.

In order to fully meet the agility and expressiveness capabilities in setting up datapaths, we claim that the orchestration process must be underpinned in a source routing approach. Therefore, in Chapter 5, we introduced a novel multipath routing, named M-PolKA (Multipath Polynomial Key-based Architecture). M-PolKA's routing mechanism is based on RNS encoding [Liberato et al. 2018], with polynomial arithmetic using GF of order 2 [Shoup 2009].

M-PolKA's approach proposed a new representation for a fully stateless multipath SR approach as a generalization of a previous work named PolKA [Dominicini et al. 2020], which was driven to single paths. Thus, we demonstrated its feasibility deploying it at a high-performance network testbed (RARE [1]) composed by a set of latest Tofino's switches [GEANT 2021] and mapping the polynomial mathematical operations implemented by reusing CRC within the programmable switches. That implementation achieved equivalent performance for packets forwarding compared to the traditional table based forwarding approaches. M-PolKA gives us more expressiveness and provides multipath capability, opening the possibility of enabling reliability functionalities in the new generation networks for any topology.

6.2 Future Works

Notwithstanding our efforts to limit the scope of this work, our research activities have inevitably touched on a diversity of other topics. Hence, the future works involve exploring the RNS-based SR properties in programmable data planes to enable a new range of modern and innovative applications, such as:

- **Traffic engineering and quality of service**, once we allow traffic engineering to make optimized decisions via exploring all the network capacities of the underlay when mapping the service overlay; we can explicitly select amongst all the existing paths and quickly modify these paths for variable demands and also load-balance the traffic according to the QoS of each application into the network. Thus, with M-PolKA we can effectively maximize network

[1] https://wiki.geant.org/display/RARE

resource utilization and alleviate all the states' maintenance across the network. As shown by Chapter 5, we did not explore all the mechanisms for traffic engineering, such as load balance;

- **Multi-layer networks and slicing,** to deliver expressiveness of M-PolKA's polynomial scheme that can extend by using GFs of higher orders for more complex routing problems or via stacked *routeID*s, which can add to the encoding *routeID* information like wavelength switching, network slicing, and traffic classification. Another aspect that we have not explored is regarding the inter-domain routing, such as using stacked *routeID*s, where each *routeID* regards a specific domain. Finally, an open issue is how to cooperate with legacy protocols, such as MPLS or GMPLS, which, for example, we can ship the *routeID* inside the MPLS protocol. The project RARE [GEANT 2021] uses the BIER label as part of the MPLS protocol;

- **Route authenticity**, brings the tampering of the *routeID* as one of the challenges for SR approaches. To solve this problem, it is possible to explore the property of *forwarding without header modifications*. Since the packet header does not change throughout all the paths, the source can sign the *routeID* information, and the nodes could verify this signature before making the forwarding decision. In our approach, taking the premise of no header modification during the forwarding process, we do not need to recalculate the decision on every hop, which eliminates many operations being performed by the switches;

- **Use of In-band Telemetry**, despite not yet supporting a full queue scheduling programmability in the P4 language [Paolucci et al. 2021], it may provide per-packet latency/jitter performance in multiple paths guaranteed by employing **in-band telemetry** (INT) and M-PolKA presented in Chapter 5. For example, we can use the M-PolKA representation to send packets into multipaths with the INT headers to collect the telemetry overall links in parallel. Therefore, P4 INT header [Cugini et al. 2019] can be added/modified all packets at each traversed node, informing the outgoing time spent across each queue onto the network. Furthermore, by analyzing the INT data on cumulated delay, indirect dynamic packet scheduling policies may be implemented (i.e., dynamic per-packet classification and priority enforcement) to minimize the jitter and latency in an E2E communication.

6.3 Final Remarks

From a broader perspective, there is an amazing list of research questions around our proposals. For example, one great challenge of modern networks is selecting paths and reacting to highly variable and demanding traffic patterns. Indeed, the orchestration process has to deliver better resource utilization, reliability, and even better experience quality (QoE). Whereas, in recent years, emerging networking languages and architectures, such as P4/PISA, have created unprecedented opportunities for rapidly prototyping disruptive solutions in programmable data planes. In addition, the potential impacts of providing cross-layer programmability aligned with the line-rate performance of commercial hardware are far-reaching with the principle of separating control- and data-planes.

Therefore, even after all these years in which we have conducted this work, cross-layer network programmability and source routing are still exciting research fields. However, various properties are missing to be explored in M-PolKA. It was shown a viable paradigm to enable and empower a wide range of network applications, which was not the case when we started with this line of research. Finally, we conclude by reaffirming our belief that the work present in this thesis provides solid guidelines and interesting insight to foster expressive and agile orchestration research and experimentation across heterogeneous resources.

Bibliography

[Alvarez et al. 2018] Alvarez, P., Slyne, F., Blumm, C., Marquez-Barja, J., DaSilva, L., and Ruffini, M. (2018). Experimental demonstration of sdn-controlled variable-rate fronthaul for converged lte-over-pon. In *Optical Fiber Communication Conference*, pages Th2A–49. Optical Society of America. 53, 59

[Ayyash et al. 2016] Ayyash, M. et al. (2016). Coexistence of wifi and lifi toward 5g: concepts, opportunities, and challenges. *IEEE Communications Magazine*, 54(2):64–71. 64

[Baranda et al. 2018] Baranda, J., Mangues-Bafalluy, J., Pascual, I., Nunez-Martinez, J., l. Cruz, J. L. D., Casellas, R., Vilalta, R., Salvat, J. X., and Turyagyenda, C. (2018). Orchestration of end-to-end network services in the 5g-crosshaul multi-domain multi-technology transport network. *IEEE Communications Magazine*, 56(7):184–191. 33

[Bayhan et al. 2018] Bayhan, S., Gür, G., and Zubow, A. (2018). The future is unlicensed: Coexistence in the unlicensed spectrum for 5g. *CoRR*. 64

[Bogue 2017] Bogue, R. (2017). Cloud robotics: a review of technologies, developments and applications. *Industrial Robot: An International Journal*, 44(1):1–5. 11, 33

[Both et al. 2019] Both, C. et al. (2019). Futebol control framework: Enabling experimentation in convergent optical, wireless, and cloud infrastructures. *IEEE Communications Magazine*, 57(10):56–62. vi, vii, viii, 12, 35, 36, 45, 47, 55, 57, 58, 60, 62, 65, 110, 122

[Boulogeorgos et al. 2018] Boulogeorgos, A. et al. (2018). Terahertz technologies to deliver optical network quality of experience in wireless systems beyond 5G. *IEEE Communications Magazine*, 56(6):144–151. 33, 44

[Bublitz et al. 2017] Bublitz, C. F. et al. (2017). Unsupervised segmentation and classification of snoring events for mobile health. In *IEEE Global Communications Conference*, pages 1–6. 57, 59

[Buccheri et al. 2018] Buccheri, L. et al. (2018). Hybrid retransmission scheme for qos-defined 5g ultra-reliable low-latency communications. In *IEEE WCNC*. 38

[Cain et al. 2002] Cain, B., Deering, D. S. E., Fenner, B., Kouvelas, I., and Thyagarajan, A. (2002). Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 3376, RFC Editor. 41

[Casado et al. 2010] Casado, M., Koponen, T., Ramanathan, R., and Shenker, S. (2010). Virtualizing the Network Forwarding Plane. In *ACM*, PRESTO '10, pages 8:1–8:6, New York, NY, USA. ACM. 20

[Cena et al. 2018] Cena, G., Scanzio, S., and Valenzano, A. (2018). A prototype implementation of wi-fi seamless redundancy with reactive duplication avoidance. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 179–186. 38, 70

[Ceravolo et al. 2018] Ceravolo, I. et al. (2018). O2CMF: Experiment-as-a-service for agile Fed4Fire deployment of programmable NFV. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3. 47

[Comer and Rastegarnia 2019] Comer, D. and Rastegarnia, A. (2019). Toward disaggregating the sdn control plane. *IEEE Communications Magazine*, 57(10):70–75. 96

[Cugini et al. 2019] Cugini, F., Gunning, P., Paolucci, F., Castoldi, P., and Lord, A. (2019). P4 in-band telemetry (int) for latency-aware vnf in metro networks. In *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. 124

[do Carmo et al. 2019a] do Carmo, A. P., Vassallo, R. F., de Queiroz, F. M., Picoreti, R., Fernandes, M. R., Gomes, R. L., Martinello, M., Dominicini, C. K., Guimarães, R., Garcia, A. S., Ribeiro, M. R. N., and Simeonidou, D. (2019a). Programmable intelligent spaces for industry 4.0: Indoor visual localization driving attocell networks. *Trans. Emerg. Telecommun. Technol.*, 30(11). 123

[do Carmo et al. 2019b] do Carmo, A. P., Vassallo, R. F., de Queiroz, F. M., Picoreti, R., Fernandes, M. R., Gomes, R. L., Martinello, M., Dominicini, C. K., Guimarães, R., Garcia, A. S., Ribeiro, M. R. N., and Simeonidou, D. (2019b). Programmable intelligent spaces for industry 4.0: Indoor visual localization driving attocell networks. *Transactions on Emerging Telecommunications Technologies*, 30(11):e3610. e3610 ett.3610. 63, 65

[Dominicini et al. 2017] Dominicini, C. K. et al. (2017). Virtphy: Fully programmable nfv orchestration architecture for edge data centers. *IEEE Transactions on Network and Service Management*, 14(4):817–830. 77

[Dominicini et al. 2020] Dominicini, C. K., Mafioletti, D. R., Locateli, A. C., Villaca, R., Martinello, M., and Ribeiro, M. N. (2020). Polka: Polynomial key-based architecture
for source routing in network fabrics. In *Proceedings of the IEEE Conference on Network Softwarization - NETSOFT '20*, pages 154–155, Ghent, Belgian. ACM Press. 14, 40, 42, 80, 82, 91, 123

[ETSI MEC ISG 2014] ETSI MEC ISG (2014). Mobile edge computing - introductory technical white paper. 26

[ETSI NFV ISG 2014] ETSI NFV ISG (2014). NFV 002 V1.2.1. Network Functions Virtualisation (NFV); Architectural Framework. 25

[Fenner et al. 2016] Fenner, B., Handley, M. J., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z. J., and Zheng, L. (2016). Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 7761, RFC Editor. 41

[Filsfils et al. 2015] Filsfils, C., Nainar, N. K., Pignataro, C., Cardona, J. C., and Francois, P. (2015). The Segment Routing Architecture. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE. 43

[Fontes et al. 2017] Fontes, R. D. R., Mahfoudi, M., Dabbous, W., Turletti, T., and Rothenberg, C. (2017). How Far Can We Go? Towards Realistic Software-Defined Wireless Networking Experiments. *Computer Journal*, 60(10):1458–1471. 37

[Fontes et al. 2015] Fontes, R. R., Afzal, S., Brito, S. H. B., Santos, M. A. S., and Rothenberg, C. E. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 384–389. 115

[Foster et al. 2020] Foster, N., McKeown, N., Rexford, J., Parulkar, G., Peterson, L., and Sunay, O. (2020). Using deep programmability to put network owners in control. *SIGCOMM Comput. Commun. Rev.*, 50(4):82–88. 13

[Fu et al. 2018] Fu, S. et al. (2018). Software defined wireline-wireless cross-networks: Framework, challenges, and prospects. *IEEE Communications Magazine*, 56(8):145–151. 11, 33, 44

[GEANT 2021] GEANT (2021). Home - rare - gÉant federated confluence. https://wiki.geant.org/display/RARE/Home. (Accessed on 08/10/2021). 123, 124

[Gilani et al. 2017] Gilani, S. M. M. et al. (2017). Mobility management in IEEE 802.11 WLAN using SDN/NFV technologies. *Eurasip Journal on Wireless Communications and Networking*, 2017(1). 37, 39

[Gomes et al. 2016] Gomes, R. R. et al. (2016). KAR: Key-for-any-route, a resilient routing system. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop*, pages 120–127. 40, 41, 119, 120

[Gowda et al. 2016] Gowda, M., Dhekne, A., and Choudhury, R. R. (2016). The Case for Robotic Wireless Networks. *Proceedings of the 25th World Wide Web Conference (WWW 2016)*, pages 1317–1327. 38

[Guedes et al. 2012] Guedes, D., Vieira, L., Vieira, M., Rodrigues, H., and Nunes, R. (2012). Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, 30(4):160–210. 21, 22, 23

[Guerzoni et al. 2017] Guerzoni, R., Vaishnavi, I., Pérez-Caparrós, D., Galis, A., Tusa, F., Monti, P., Sganbelluri, A., Biczók, G., Sonkoly, B., Toka, L., Ramos, A., Melian, J., Dugeon, O., Cugini, F., Martini, B., Iovanna, P., Giuliani, G., de Oliveira Figueiredo, R., Murillo, L. M. C., Bernardos, C. J., Santana, C., and Szabó, R. (2017). Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey. *Trans. Emerg. Telecommun. Technol.*, 28. vi, 34

[Guimaraes S et al. 2020] Guimaraes S, R., Martinez M G, V., Mello, R. C., Mafioletti, D. R., Martinello, M., and Ribeiro, M. R. N. (2020). An SDN-NFV orchestration for reliable and low latency mobility in Off-the-Shelf WiFi. In *2020 IEEE International Conference on Communications (ICC): Mobile and Wireless Networks Symposium (IEEE ICC'20 - MWN Symposium)*, Dublin, Ireland. 63, 65, 81, 118, 119

[GÉANT 2021] GÉANT (2021). https://wiki.geant.org/pages/viewpage.action?pageId=148085131. (Accessed on 02/03/2021). vii, 100

[Han et al. 2015] Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *Communications Magazine, IEEE*, 53(2):90–97. vi, 24, 25

[Heller et al. 2012] Heller, B., Sherwood, R., and Mckeown, N. (2012). The controller placement problem. *Computer Communication Review*, 42(4):473–478. 14

[Huang et al. 2017] Huang, T. et al. (2017). A survey on large-scale software defined networking SDN testbeds: Approaches and challenges. *IEEE Communications Surveys Tutorials*, 19(2):891–917. 36

[ITU-R Recommendation 2015] ITU-R Recommendation (2015). ITU-R M.2083 IMT Vision: Framework and overall objectives of the future development of IMT for 2020 and beyond. 29

[Jin et al. 2016] Jin, X. et al. (2016). Your data center switch is trying too hard. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research*, pages 12:1–12:6, New York, NY, USA. ACM. 87

[Jyothi et al. 2015] Jyothi, S. A. et al. (2015). Towards a flexible data center fabric with source routing. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research*, pages 10:1–10:8, New York, NY, USA. ACM. 81

[Kassar et al. 2008] Kassar, M., Kervella, B., and Pujolle, G. (2008). An overview of vertical handover decision strategies in heterogeneous wireless networks. *Computer Communications*, 31:2607–2620. 117

[Komolafe 2017] Komolafe, O. (2017). IP multicast in virtualized data centers: Challenges and opportunities. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 407–413. IEEE. 81

[Lantz et al. 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *ACM*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM. 20

[Lee et al. 2007] Lee, P. P. C., Misra, V., and Rubenstein, D. (2007). Distributed algorithms for secure multipath routing in attack-resistant networks. *IEEE/ACM Transactions on Networking*, 15(6):1490–1501. 79

[Li et al. 2017] Li, X., Samaka, M., Chan, H. A., Bhamare, D., Gupta, L., Guo, C., and Jain, R. (2017). Network slicing for 5g: Challenges and opportunities. *IEEE Internet Computing*, 21(5):20–27. 32

[Li et al. 2018] Li, Z. et al. (2018). 5g urllc: Design challenges and system concepts. In *15th International Symposium on Wireless Communication Systems*. 30, 65

[Liberato et al. 2018] Liberato, A. et al. (2018). Rdna: Residue-defined networking architecture enabling ultra-reliable low-latency datacenters. *IEEE Transactions on Network and Service Management*, 15(4). 40, 69, 77, 86, 123

[Liu et al. 2019] Liu, Q. et al. (2019). Proactive mobility management based on virtual cells in sdn-enabled ultra-dense networks. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. 39

[Lucas-Estañ et al. 2018] Lucas-Estañ, M. C. et al. (2018). An experimental evaluation of redundancy in industrial wireless communications. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1075–1078. 39

[Mafioletti et al. 2020] Mafioletti, D. R., Dominicini, C. K., Martinello, M., Ribeiro, M. R. N., and d. S. Villaça, R. (2020). Piaffe: A place-as-you-go in-network framework for flexible embedding of vnfs. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6. 113

[Mao et al. 2017] Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358. 27

[Marques et al. 2017] Marques, P. et al. (2017). Experiments Overview of the EU-Brazil FUTEBOL Project. In *European Conference on Networks and Communications*, pages 1–2. 36

[Martinello et al. 2014] Martinello, M. et al. (2014). Keyflow: a prototype for evolving sdn toward core network fabrics. *IEEE Network*, 28(2):12–19. 40, 41, 77, 82

[Martínez 2019] Martínez, V. M. G. (2019). *WIFI MOBILE ACCESS WITH SOFTWARE DEFINED MULTI-CONNECTIVITY*. dissertation, Universidade Federal do Espirito Santo. vi, 30

[Martins et al. 2014] Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. (2014). ClickOS and the Art of Network Function Virtualization. In *NSDI*, NSDI'14, pages 459–473, Berkeley, CA, USA. USENIX Association. 24

[Martínez et al. 2018] Martínez, V. M. G. et al. (2018). Ultra reliable communication for robot mobility enabled by sdn splitting of wifi functions. In *2018 IEEE Symposium on Computers and Communications (ISCC)*. vi, 39, 63, 66

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69. vi, 20, 21, 22

[Mello et al. 2019] Mello, R. et al. (2019). On human-in-the-loop cps in healthcare: A cloud-enabled mobility assistance service. *Robotica*, page 1–17. 65

[Merling et al. 2018] Merling, D., Menth, M., Warnke, N., and Eckert, T. (2018). An overview of bit index explicit replication (bier) – ietf journal. *IETF*. vi, 43

[Mininet 2018] Mininet (2018). An Instant Virtual Network on your Laptop (or other PC). 98

[Mishra et al. 2003] Mishra, A., Shin, M., and Arbaugh, W. (2003). An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *ACM SIGCOMM Computer . . .*, 33(2):93–102. 37

[Moura et al. 2015] Moura, H. et al. (2015). Ethanol: Software defined networking for 802.11 wireless networks. In *IFIP/IEEE International Symposium on Integrated Network Management*, pages 388–396. 37, 56, 58

[Nencioni et al. 2018] Nencioni, G., Garroppo, R. G., Gonzalez, A. J., Helvik, B. E., and Procissi, G. (2018). Orchestration and Control in Software-Defined 5G Networks: Research Challenges. *Wireless Communications and Mobile Computing*, 2018:1–18. vi, 33, 36

[Networks 2017] Networks, I. (2017). P4d2 – programmable data plane at terabit speeds. vii, 102

[Nielsen et al. 2018] Nielsen, J. J., Liu, R., and Popovski, P. (2018). Ultra-reliable low latency communication using interface diversity. *IEEE Transactions on Communications*, 66(3):1322–1334. 39, 65, 78

[Nikaein et al. 2014] Nikaein, N. et al. (2014). OpenAirInterface: an open LTE network in a PC. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 305–308. ACM. 56, 58

[Open Networking Foundation 2014] Open Networking Foundation (2014). OpenFlow-enabled SDN and Network Functions Virtualization. 11, 20

[Openvswitch.org 2015] Openvswitch.org (2015). What is Open vSwitch? vi, 23

[P4 2017] P4 (2017). The p4 14 language specification. https://p4.org/p4-spec/p4-14/v1.0.4/tex/p4.pdf. vii, 93

[P4 2018] P4 (2018). P4~16~ portable switch architecture (psa). https://p4.org/p4-spec/docs/PSA-v1.1.0.html. 93, 101

[P4 v16 2020] P4 v16 (2020). P4 16 Language Specification version 1.0.0. vi, 24, 87, 91, 100

[Paolucci et al. 2021] Paolucci, F., Cugini, F., Castoldi, P., and Osinski, T. (2021). Enhancing 5g sdn/nfv edge with p4 data plane programmability. *IEEE Network*, pages 1–7. 124

[Peterson and Brown 1961] Peterson, W. W. and Brown, D. T. (1961). Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235. 91

[Pocovi et al. 2018] Pocovi, G., Shariatmadari, H., Berardinelli, G., Pedersen, K., Steiner, J., and Li, Z. (2018). Achieving ultra-reliable low-latency communications: Challenges and envisioned system enhancements. *IEEE Network*, 32(2):8–15. 38, 64

[Popovski et al. 2018] Popovski, P., Nielsen, J. J., Stefanovic, C., d. Carvalho, E., Strom, E., Trillingsgaard, K. F., Bana, A., Kim, D. M., Kotaba, R., Park, J., and Sorensen, R. B. (2018). Wireless access for ultra-reliable low-latency communication: Principles and building blocks. *IEEE Network*, 32(2):16–23. 38

[Raghavendra et al. 2007] Raghavendra, R., Belding, E. M., Papagiannaki, K., and Almeroth, K. C. (2007). Understanding Handoffs in Large IEEE 802 . 11 Wireless. *Wireless Networks*, pages 333–338. 37

[Ramos et al. 2013] Ramos, R. M., Martinello, M., and Rothenberg, C. E. (2013). Slickflow: Resilient source routing in data center networks unlocked by openflow. In *38th Annual IEEE Conference on Local Computer Networks, Sydney, Australia, October 21-24, 2013*, pages 606–613. IEEE Computer Society. 78

[Reed et al. 2016] Reed, M. J., Al-Naday, M., Thomos, N., Trossen, D., Petropoulos, G., and Spirou, S. (2016). Stateless multicast switching in software defined networks. In *2016 IEEE International Conference on Communications, ICC 2016*, pages 1–7. IEEE. 40, 41

[Ren et al. 2017] Ren, Y. et al. (2017). Flowtable-free routing for data center networks: A software-defined approach. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6. 86

[Rentschler and Laukemann 2012] Rentschler, M. and Laukemann, P. (2012). Towards a reliable parallel redundant wlan black channel. In *2012 9th IEEE International Workshop on Factory Communication Systems*, pages 255–264. 38, 70

[Rosa et al. 2015] Rosa, R. V., Santos, M. A. S., and Rothenberg, C. E. (2015). Md2-nfv: The case for multi-domain distributed network functions virtualization. In *2015 International Conference and Workshops on Networked Systems (NetSys)*, pages 1–5. 29

[Routray et al. 2013] Routray, S. K. et al. (2013). Statistical model for link lengths in optical transport networks. *J. Opt. Commun. Netw.*, 5(7):762–773. 86

[Ryu 2015] Ryu (2015). Ryu SDN Framework Community. 50, 53, 72

[Sanchez et al. 2016] Sanchez, M. I., De La Oliva, A., and Mancuso, V. (2016). Experimental evaluation of an SDN-based distributed mobility management solution. *2016 ACM Workshop on Mobility in the Evolving Internet Architecture, MobiArch 2016*, 03-07-Octo. 37

[Saraiva de Sousa et al. 2019] Saraiva de Sousa, N. F., Lachos Perez, D. A., Rosa, R. V., Santos, M. A., and Esteve Rothenberg, C. (2019). Network Service Orchestration: A survey. *Computer Communications*, 142-143(May):69–94. vi, 11, 12, 23, 27, 28

[Satyanarayanan 2017] Satyanarayanan, M. (2017). The Emergence of Edge Computing. *Computer*, 50(1):30–39. 26

[Schroeder 2009] Schroeder, M. (2009). *Number theory in science and communication: with applications in cryptography, physics, digital information, computing, and self-similarity*. Springer, Berlin, Heidelberg. 42

[Shahbaz et al. 2019] Shahbaz, M. et al. (2019). Elmo. In *Proceedings of the ACM Special Interest Group on Data Communication - SIGCOMM '19*, pages 458–471, New York, New York, USA. ACM Press. 40, 41, 69, 86

[Sherry et al. 2012] Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. (2012). Making Middleboxes Someone else's Problem: Network Processing As a Cloud Service. In *SIGCOMM*, SIGCOMM '12, pages 13–24, New York, NY, USA. ACM. 24

[Sherwood et al. 2009] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2009). FlowVisor: A Network Virtualization Layer. *OpenFlow Switch Consortium, Tech. Rep.* vi, 21

[Shi and Dustdar 2016] Shi, W. and Dustdar, S. (2016). The Promise of Edge Computing. *Computer*, 49(5):78–81. vi, 27

[Shoup 2009] Shoup, V. (2009). *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press. 14, 82, 85, 123

[Singh et al. 2015] Singh, S. K. et al. (2015). A survey on internet multipath routing and provisioning. *IEEE Communications Surveys Tutorials*, 17(4):2157–2175. 78, 79

[Slyne et al. 2019] Slyne, F., Guimaraes, R. S., and Zhang, Y. (2019). Coordinated fibre and wireless spectrum allocation in SDN-controlled wireless-optical-cloud converged architecture. In *Optical Fiber Communication Conference 2019*. IEEE. vii, viii, 51, 52, 53, 54, 55

[Soliman et al. 2012] Soliman, M., Nandy, B., Lambadaris, I., and Ashwood-Smith, P. (2012). Source routed forwarding with software defined control, considerations and implications. *CoNEXT Student 2012 - Proceedings of the ACM Conference on the 2012 CoNEXT Student Workshop*, pages 43–44. 14, 43

[Suer et al. 2020] Suer, M. et al. (2020). Multi-connectivity as an enabler for reliable low latency communications—an overview. *IEEE Communications Surveys Tutorials*, 22(1):156–169. 78, 79, 80, 81

[Sunshine 1977] Sunshine, C. A. (1977). Source routing in computer networks. *SIGCOMM Comput. Commun. Rev.*, 7(1):29–33. 43

[Suresh et al. 2012] Suresh, L., Schulz-Zander, J., Merz, R., Feldmann, A., and Vazao, T. (2012). Towards programmable enterprise WLANS with Odin. *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, page 115. 37

[Systems 2017] Systems, S. R. (2017). srsLTE Library https://github.com/srslte. 51

[Tzanakaki et al. 2017] Tzanakaki, A. et al. (2017). Wireless-optical network convergence: Enabling the 5G architecture to support operational and end-user services. *IEEE Communications Magazine*, 55(10):184–192. 12

[Verdi et al. 2010] Verdi, F. L., Rothenberg, C. E., Pasquini, R., and Magalhães, M. (2010). Novas arquiteturas de data center para cloud computing. *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-Gramado-RS*. 21, 23

[Wessing et al. 2002] Wessing, H., Christiansen, H., Fjelde, T., and Dittmann, L. (2002). Novel Scheme for Packet Forwarding Without Header Modifications in Optical Networks. *J. Lightwave Technol.*, 20(8):1277. 41, 82, 87

[Wijnands et al. 2017] Wijnands, I., Rosen, E. C., Dolganow, A., Przygienda, T., and Aldrin, S. (2017). Multicast Using Bit Index Explicit Replication (BIER). RFC 8279. 40, 42, 86

[Xu et al. 2017] Xu, C. et al. (2017). A novel multipath-transmission supported software defined wireless network architecture. *IEEE Access*, pages 2111–2125. 39

[Yang and Cheng 2019] Yang, H. and Cheng, L. (2019). Bounding network-induced delays of wireless prp infrastructure for industrial control systems. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 38

[Yousaf et al. 2017] Yousaf, F. Z., Bredel, M., Schaller, S., and Schneider, F. (2017). NFV and SDN-Key technology enablers for 5G networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478. vi, 31, 32, 44

[Zehl et al. 2016] Zehl, S., Zubow, A., and Wolisz, A. (2016). BIGAP - A seamless handover scheme for high performance enterprise IEEE 802.11 networks. *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 1(Noms):1015–1016. 37

[Zeljković et al. 2018] Zeljković, E. et al. (2018). Proactive access point driven handovers in ieee 802.11 networks. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 261–267. 39

[Zeljković et al. 2019] Zeljković, E. et al. (2019). Abraham: Machine learning backed proactive handover algorithm using sdn. *IEEE Transactions on Network and Service Management*, 16(4):1522–1536. 39