Towards a flexible Network Operating System Testbed for the Computing Continuum

Julien Caposiena, Oscar Carrillo, Frédéric Le Mouël, Baptiste Jonglez, Pierre Neyron, Thierry Arrabal

Towards a flexible Network Operating System Testbed for the Computing Continuum

Julien Caposiena
INSA Lyon, Inria, CITI, UR3720
69621 Villeurbanne, France
julien.caposiena@insa-lyon.fr

Oscar Carrillo
CPE Lyon, INSA Lyon, Inria, CITI, UR3720
69621 Villeurbanne, France
oscar.carrillo@insa-lyon.fr

Frédéric Le Mouël
INSA Lyon, Inria, CITI, UR3720
69621 Villeurbanne, France
frederic.le-mouel@insa-lyon.fr

Baptiste Jonglez
Nantes Université, École Centrale Nantes, CNRS, Inria, LS2N, UMR 6004
F-44000 Nantes, France
baptiste.jonglez@inria.fr

Pierre Neyron
CNRS, Univ. Grenoble Alpes, Inria, Grenoble INP, LIG, UMR 5217
38058 Grenoble, France
pierre.neyron@imag.fr

Thierry Arrabal
INSA Lyon, Inria, CITI, UR3720
69621 Villeurbanne, France
thierry.arrabal@insa-lyon.fr

***Abstract*—In recent years, emerging computing paradigms have paved the way for the development of the Computing Continuum, a concept that enables applications to efficiently allocate geo-distributed resources across the network. Despite its potential, fully realizing the Computing Continuum remains a challenge due to the lack of suitable research infrastructures that support experimentation at scale.**

To address this gap, we propose a conceptual testbed designed to enhance the replicability, scalability, and robustness of Computing Continuum and network operating system experiments. Our testbed provides a high degree of flexibility by allowing experimenters to modify the operating systems of network equipment and dynamically reconfigure network topologies. To illustrate its versatility, we define three distinct usage scenarios, ranging from multi-operator environments to internal network architectures within telecommunication providers, all of which can be deployed on the proposed network topology. Additionally, this article explores solutions for virtual topology management, operating system deployment, and service orchestration.

***Index Terms*—Continuum Computing, Network Operating System, Testbed, Network Equipment, Edge-to-Cloud**

I. Introduction

Over the years, testbeds have evolved to fulfil more and more scientific experiment needs. Some are evolving towards specific demands such as Software Defined Networking, High Performance Computing, Big Data, Artificial Intelligence, Cognitive Radio Networks, among others. While others tend to be more generic, like for testing one or more entire computing tiers including Cloud, Fog, Mist, Edge and Extreme-Edge. Thus, the features regarding modern testbeds are rather scattered, often leading to problem like the testbed being too specific or too generic for certain experiments.

Following scientific progress, solutions of merging all the computing tiers are appearing like the so-called "Computing Continuum". The Computing Continuum refers to a distributed computing model allowing end-to-end resource orchestration that considers computation, storage and network for applications executed across the Edge, Fog, and Cloud computing tiers [1], [2]. This paradigm ensures communication and open availability between geographically distributed computing resources [3]. The Computing Continuum often manifests in large scale distributed applications that involve most of the computing tiers, such as an Internet Service Provider or a video service.

When experimenting with the Computing Continuum in a scientific context, one or more large-scale testbeds are often needed. As the continuum may implement any computing tier from Extreme-Edge to Cloud, the testbed network topology ought to be flexible and highly reconfigurable. Regarding certain edge-cases, the likelihood of the network topology necessitating to shrink or expand over time may be high, thus requiring both vertical and horizontal scalability. Following the likely evolving network topology needs, highly heterogeneous equipments are needed in order to fulfil most demands. To attain this level of flexibility, experimenters must have the ability to change, modify, and reconfigure the operating systems of most testbed components, including routers and switches. Although many Network Operating Systems (NOS) have emerged, the capability to dynamically replace the bare-metal operating system of network equipment remains scarce.

This article thus presents the following contributions:

- Three Computing Continuum usage scenarios, detailed in Section II.
- A comprehensive review of related work on network operating systems, research infrastructures, and network topologies in Section III.
- A testbed designed for Edge-to-Cloud experiments, including the implementation of a Computing Continuum. It features: A physical network topology described in Section IV.A ; Dynamic reconfiguration of the virtual network topology with tooling proposition and examples, discussed in Section IV.B ; The ability to replace the operating systems of network equipment, with suggested deployment solutions in Section IV.C ; Service orchestration across the testbed, as outlined in Section IV.D.

II. Usage scenarios

To assess the feasibility and applicability of network equipment operating systems in the edge-to-cloud continuum, we propose three usage scenarios that represent challenges and opportunities in contemporary networking environments.

A. Multi-operator

This scenario explores the inter-connectivity between Local Area Networks (LANs), Metropolitan Area Networks (MANs), and Wide Area Networks (WANs) managed by multiple operators. Such environments are inherently complex due to differing administrative policies, protocols, and operational objectives.

Key elements of this scenario include the implementation and testing of inter-domain routing protocols, such as the Border Gateway Protocol (BGP). BGP is critical for maintaining global internet connectivity by enabling the exchange of routing information between autonomous systems. Additionally, the use of Multi-Protocol Label Switching (MPLS) is examined for its ability to enhance traffic engineering, optimize data flows, and ensure quality of service (QoS) across interconnected networks. By addressing these elements, this scenario aims to simulate the challenges and solutions for achieving seamless integration and collaboration in multi-operator settings.

B. Internet Service Provider to Telecommunications Service Provider

The scenario aims to simulate the complexities of cross-provider collaboration, particularly in environments where stringent requirements for QoS and data integrity are paramount. Telemedicine, for instance, requires real-time data transmission to ensure effective remote diagnostics and treatment. Similarly, industrial automation relies on synchronized communication to maintain operational efficiency. This scenario investigates how ISPs and TSPs can align their infrastructures, protocols, and service-level agreements (SLAs) to meet these demands while minimizing latency and maximizing reliability.

C. Internet Service Provider Internal Network

This scenario examines the internal operations of an ISP's network, which must balance scalability, performance, and resilience to meet ever-growing user demands. ISPs face unique challenges in optimizing their internal topologies to handle high volumes of traffic while maintaining fault tolerance and service continuity.

The scenario aims to emulate the intricate dynamics of ISP networks, including internal routing, resource allocation, and failure recovery mechanisms. A particular focus is placed on optimizing traffic flows to prevent bottlenecks and ensure equitable distribution of bandwidth across the network. Additionally, the scenario considers the integration of emerging technologies to enhance network management capabilities and enable proactive responses to potential issues. By addressing these aspects, this scenario provides a framework for evaluating the adaptability and robustness of ISP infrastructure in the face of evolving demands and technological advancements.

III. Related work

This section describes existing testbeds, related hardware, network operating systems and defines network topology regarding the outlined usage scenarios.

A. Existing infrastructures

The two last decades have seen the rise of many research infrastructures. These research infrastructures allow the conduct of various computer science experiments. As visible from Table I, we have listed many popular or emergent research infrastructures. They will be compared in order to determine which one is able to host our proposed testbed. As they are not all intended for the same usage, the computing tiers involved may differ. For example, some may be rather specific for telecom and wireless experiments like CorteXlab, whereas others tend to provide a more generic network infrastructure like Grid'5000.

TABLE I: Research infrastructures

Infrastructure name	Computing tier(s) involved	Launch year	Current status
SLICES-RI [4]	Edge-to-Cloud	2022	Partially active
EdgeNet[1] [5]	Edge-to-Cloud	2019	Active
Emulab [6]	Cloud/Edge	2002	Active
Chameleon Cloud [7]	Cloud	2015	Active
Grid'5000[1] [8]	Cloud	2006	Active
FogGuru [9]	Fog	2020	No news since 2023
YOUPI[1] [10]	Fog/Edge	2019	Active
CorteXlab[1] [11]	Edge	2014	Active

SLICES-RI is an ambitious European project, that is an ESFRI project since 2021. It aims to reassemble major research infrastructure across Europe.

The EdgeNet testbed relies on Kubernetes for its infrastructure. As such, it only allows experimenters to deploy containers: while this allows a good variety of Linux OS to be deployed, it does not give access to the bare-metal hardware, and no NOS images are available.

In the Emulab testbed, users are not allowed to modify the bare-metal network operating systems of routers and switches. However, they can change and reorganize the virtual network topology using Emulab's topology description tools.

Grid'5000 grants experimenters the ability to use and modify any OS on heterogeneous physical machines, thanks to Kadeploy, its OS provisioning service. Kadeploy gives full control on the bare-metal installations of clusters of machines, up to actions requiring access to serial consoles. A work is in progress to add NOS compatibility, allowing to automatically provision the OS of a cluster of white box switches. The project also includes full-mesh connectivity between 8 Tofino white box switches on the Strasbourg site, allowing the experimenter to build any virtual network topology [12].

[1]Are precursors of the "SLICES-FR" French node of SLICES-RI.

Chameleon Cloud offers bare-metal access to physical machines like Grid'5000, but provides a limited list of OSs to use: no NOS is part of it. Experimenters can create isolated networks from their WebUI.

In between Cloud and Edge computing, FogGuru runs on Raspberry Pis that are not intended to act as network equipments. In addition, like EdgeNet, FogGuru uses Kubernetes, thus the images are not running bare-metal.

Like Grid'5000, YOUPI is very permissive by granting full permissions for experimenters to modify the machines. Therefore, allowing to change for any Linux-based OS, including most NOSs. Along 2025, the platform may evolve toward a dynamically reconfigurable network for experimenters.

On the edge computing side, CorteXlab is exclusively oriented toward wireless communication and does not offer the possibility to the experimenter to modify the network virtual topology too. Also, since the machines use Docker, the images put on are not running bare-metal.

TABLE II: Existing research infrastructures abilities

Infrastructure name	Ability to modify network operating systems bare-metal	Ability to change and reorganize the virtual network topology
EdgeNet	No NOS and not bare-metal[2]	No
Emulab	No NOS images available	Yes
Chameleon Cloud	No NOS images available	Yes
Grid'5000	WIP on the Strasbourg site	Yes
FogGuru	No NOS and not bare-metal[2]	No
YOUPI	Yes	Planned along 2025
CorteXlab	No NOS and not bare-metal[2]	Wireless network only

As summarized within Table II, the testbed that fits best what we are trying to achieve is Grid'5000. It allows experimenters to change machines' operating systems with whatever they need, and provides the KaVLAN [13] services which allows on-demand configuration of VLANs between nodes. With the upcoming integration of white box switches in the Strasbourg site, it will allow experimenters the capability to reconfigure the Layer 2 (L2) network topology between a set of both experimental network switches and servers.

The proposed testbed extends Grid'5000 capabilities by ensuring the possibility to implement and test most of the Computing Continuum scenarios as presented in Section II. Thus allowing repeatable, replicable and reproducible experiments within the extensively documented Grid'5000.

B. Existing Hardware and Operating Systems

In this section, we review the predominant hardware platforms and network operating systems used in research and industry, highlighting their relevance to Computing Continuum experimentation.

a) *Hardware:*

Routers and switches serve distinct roles. L2 switches forward frames via MAC addresses within a broadcast domain, while L3 switches add IP routing with hardware acceleration (ASIC, TCAM) for fast inter-VLAN communication. Routers manage WAN, complex policies, and protocol-based routing (OSPF, BGP, MPLS) using CPU-driven packet processing for NAT and VPNs. L3 switches approach router performance but focus on local, high-throughput environments. The following hardware list is thus divided into routers and switches (L2, L3).

Selecting suitable network equipment for the proposed usage scenarios requires hardware that supports operating system flexibility, Time-Sensitive Networking (TSN) capabilities, and Software-Defined Radio (SDR) integration. Several white-box[3] switches and routers meet these criteria.

For white-box switches, models such as the **Edgecore AS7326-56X** and the **Dell Z9264F-ON** offer high-speed switching, and TSN support for deterministic networking. On the router side, platforms like the **UfiSpace S9500-30XS** and **Nokia 7220 IXR-D2** provide open networking capabilities, allowing the deployment of customizable operating systems suited for advanced research.

b) *Operating System:*

A wide variety of network operating systems are available: Table III lists the most popular ones.

TABLE III: Network Operating Systems

Operating System	Vendor	ASIC vendor compatibility	Target	License
OpenWrt	OpenWrt Project	Many	Router, Switch	Free, GPL-2.0
VyOS	VyOS Networks	Many	Router	Free, GPL-2.0
SONiC OS	Microsoft, SONiC Foundation	Many	Router, Switch	Free, GPL-2.0
BSDRP	BSD Router Project	No	Router	Free, BSD-2-Clause
AlliedWare Plus	Allied Telesis	Allied Telesis only	Router, Switch	Paying license
Router OS	MikroTik	Many	Router	Paying license
SwOS	MikroTik	MikroTik only	Switch	Paying license
OS10	DELL	DELL only	Switch	Paying license
DNOS9 (formerly FTOS)	DELL	DELL only	Switch	Paying license
ExtremeXOS (Switch Engine)	Extreme Networks	Extreme Networks only	Switch	Paying license
EOS	Arista	Arista only	Switch	Paying license
PicOS	Pica8 Software Inc.	Many	Switch	Paying license
Cumulus Linux	Cumulus Networks, NVIDIA	Many	Switch	Paying license
Junos OS	Juniper Networks	Juniper Networks only	Router, Switch	Paying license
iOS XE	CISCO	CISCO only	Router, Switch	Paying license

[2]Kubernetes/Docker images only

[3]Network equipment that is assembled from standardized commodity parts

A key factor in addressing the problem we aim to solve is the ability to modify the operating system of network equipment. Consequently, hardware compatibility plays a crucial role. The proposed testbed should support a broad range of architectures, including both standard x86-based platforms and white box switches, while maintaining compatibility with ASIC-based hardware. The more an operating system supports diverse ASIC vendors, the more versatile and valuable it becomes for experimentation. This flexibility ensures that the testbed can accommodate real-world deployment scenarios where proprietary ASICs and merchant silicon coexist. Additionally, the choice of operating system—whether proprietary or open-source—remains at the discretion of the experimenter, allowing for a variety of research and industry use cases.

C. *Existing network architectures*

This section focuses on the network architectures linked to the mentioned usage scenarios in Section II. Each emphasizes realizable experiments within the proposed testbed.

Regarding the first usage scenario, multi-operator communication, end-to-end experimenter communication that traverses the internet must be achievable. The average packet takes about 4.06 Autonomous Systems (AS) path length[4] to go through the internet [14]. In this work, we are going to focus on a minimal AS network topology that involves four meshed routers. Consequently, allowing to reproduce the average route to traverse the internet. Fig. 1 [15] illustrates a typical ISP network topology.

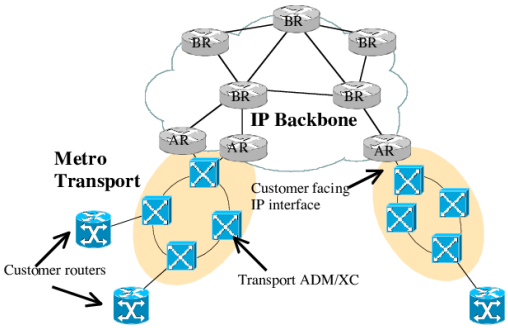

Fig. 1: Typical ISP Network Topology

The second usage scenario focuses on ISP to TSP communication, thus allowing end-to-end experimenter communication. Fig. 2 from [16] emphases the need of telecommunication hardware by connecting internet to a VoLTE network architecture. This can be achievable by the use of SDR cards or by interconnecting others research infrastructures such as CorteXlab or IoT-LAB [17].

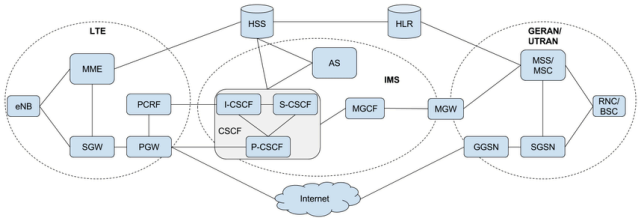

Fig. 2: Core network architecture serving Voice over LTE - including 2G, 3G, 4G and IMS components

The third usage scenario, focusing on an internet service provider's internal network, prioritizes high performance, scalability, and robustness. This environment enables experimentation with custom communication protocols, fully distributed or fluid computing architectures, and TSN, among other advanced networking paradigms, to optimize efficiency and reliability. The network topology remains flexible, allowing for adaptation based on specific research and operational requirements.

A promising direction for future work is the integration of Deep-Generative Graphs for the Internet (DGGI) [18], to generate realistic network topologies for our usage scenarios. Traditional topology generators often fail to capture key structural properties such as betweenness, clustering, and assortativity, which are crucial for accurately modeling intra-AS networks. By leveraging DGGI, we could create synthetic topologies that closely resemble real-world networks, enhancing the validity of our testbed experiments. This approach would enable more data-driven topology selection, improving the evaluation of routing protocols, network OS behavior, and scalability under diverse conditions.

IV. Conceptual Framework

Designing an effective testbed for Computing Continuum experiments requires a scalable and sustainable framework that supports flexibility, reproducibility, and long-term adaptability. This section outlines the key design principles guiding our approach, ensuring that the testbed can accommodate diverse network environments and evolving research needs. Additionally, we propose a structured workflow for experimentation, detailing how network operating systems can be integrated and deployed within the testbed to facilitate rigorous and repeatable testing.

A. *Physical topology*

The foundation of our testbed's physical topology is inspired by the Grid'5000 project in the Strasbourg site to support core-network experiments. However, certain limitations must be addressed to better support telecommunications-oriented experiments and custom communication protocols as seen with the usage scenarios.

[4]Number of autonomous systems a data packet traverses between its source and destination

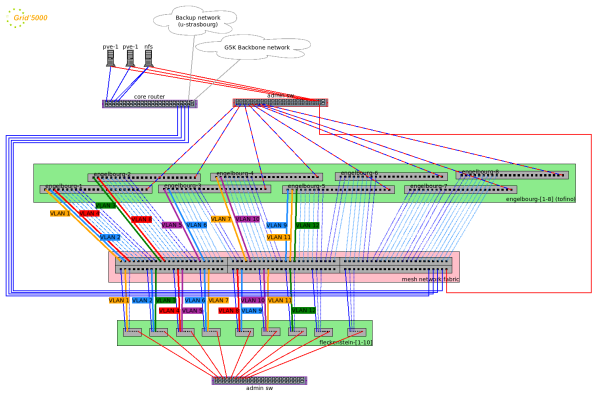

Fig. 3: Grid'5000 Strasbourg site network topology [12]

The network topology of the Grid'5000 Strasbourg site visible in Fig. 3 [12] consists of a core routing infrastructure interconnected to other Grid'5000 sites, providing high-speed connectivity for distributed computing experiments. The compute and networking experimental resources are organized into distinct groups, notably the `engelbourg` nodes, which consist of programmable Tofino-based white box switches, and the `fleckenstein` nodes, which are standard HPE servers with several network interfaces. A dedicated mesh network fabric interconnects these resources, facilitating flexible networking configurations thanks to L2 VLAN switching. VLAN segmentation is extensively used, with multiple VLANs ensuring logical isolation and controlled data flow across the infrastructure.

This design allows many different virtual topologies to be setup by the experimenter, thanks to the mesh network fabric. However, it comes at a cost: all network traffic between the resources must go through this L2 fabric, preventing the use of advanced or experimental layer-2 protocols. Examples of advanced L2 usage include VLAN stacking (802.1ad), custom QoS policies, or using the Precision Time Protocol (PTP). Another constraint is the lack of end-device diversity, particularly for telecommunications research, where real-world end-user behavior and heterogeneous device interactions play a crucial role.

To overcome these challenges, we propose direct interconnection of experimenter-controlled routers within the testbed, reducing dependency on infrastructure switches and enabling the formation of minimal AS-like virtual networks. This design allows for greater control over packet forwarding, facilitates custom protocol deployment, and enhances scalability for multi-domain networking experiments. However, this constrains our testbed to single-user experiments: as soon as an experimenter has full control over one network device (e.g. a L2/L3 switch in Fig. 4), no other experimenter can safely use the other devices because their connectivity directly depend on each other.

A second key enhancement involves integrating Software-Defined Radio (SDR) capabilities by equipping the testbed with PCIe SDR cards or establishing interconnections with external research infrastructures like CorteXlab. This addition would enable the evaluation of wireless-networking scenarios, edge-to-cloud interactions, and hybrid wired-wireless architectures, further expanding the scope of possible experiments.

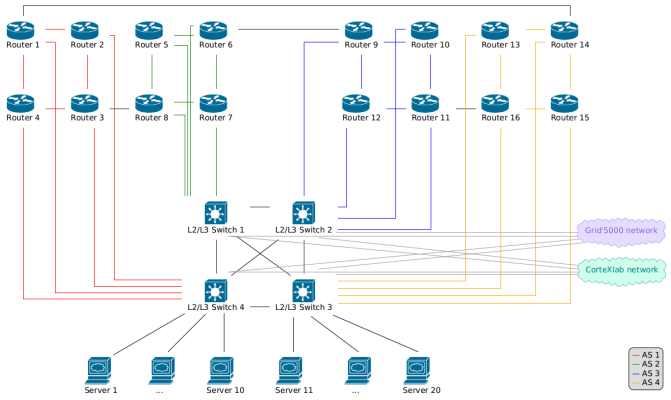

Fig. 4: Proposed network architecture

The presented network topology in Fig. 4 is structured around a multi-router, multi-switch design, facilitating flexible and scalable networking experiments. It consists of 16 routers, which are categorized into four distinct subnetworks (minimalist ASs), each represented by different colors (red, green, blue, and orange) for readability. These subnetworks are interconnected via four L2/L3 switches, forming a hierarchical structure that ensures high connectivity while maintaining isolation between different experimental setups.

The Grid'5000 network (Purple) and the CorteXlab network (Teal) as an example are integrated within the testbed, allowing for external connectivity and interaction with broader research infrastructures. The routers within each subnetwork are directly linked to their respective L2/L3 switches, which are in turn interconnected to enable communication across different domains. Additionally, each core switch connects to a pool of servers, which serve as end-hosts for various networking experiments. To ensure the usage of network topologies tools like KaVLAN, as described in Section IV.B, the core switches are P4 compatible. It is up to the experimenter to maintain P4 compatibility within the switches operating systems to benefit from the ease of network configuration offered by tools like KaVLAN. Furthermore, each device is connected to an administration network (not depicted in the diagram), which facilitates operating system changes and basic administrative tasks, also ensuring that no connection is lost with incorrect network configurations.

The topology enables routing experiments by supporting direct router-to-router connections, allowing the implementation of AS-like environments. The integration of multiple research infrastructures ensures that the testbed is suitable for testing custom communication protocols, software-defined networking (SDN) approaches, and time-sensitive networking scenarios. This design avoids dependence on a central switch, providing flexibility in defining network policies and improving resilience against potential failures.

B. Virtual topology tools

Conceptually, setting up a virtual topology for an experimentation requires three phases: 1) **Defining the desired virtual**

topology: The experimenter specifies the intended network topology using a sufficiently expressive language to accommodate a wide range of experimental setups. 2) **Mapping the virtual topology onto the physical infrastructure**: This process can be handled either by the testbed itself—leveraging tools such as `assign` from Emulab—or by the experimenter, who selects the most suitable physical resources. 3) **Configuring physical resources to establish the topology**: The testbed automates this step using internal tools, which allocate and configure physical nodes while deploying VLANs to establish the necessary network links.

a) *Virtual topology description:*

The experimenter must be able to describe a target topology as a graph. More precisely, a bipartite graph offers a general representation: the first class of graph nodes represents reconfigurable *resources* (servers or network devices), while the second class of graph nodes represents *networks*. A link in the graph represents the attachment of a *resource* (through one of its network interface cards) to a *network*. All *resources* that are attached to the same *network* can communicate with each other. Fig. 5 gives an example topology with three user-controlled *resources*: two servers and one switch that interconnects them. This topology needs two *networks* to represent the connectivity between the *resources*.

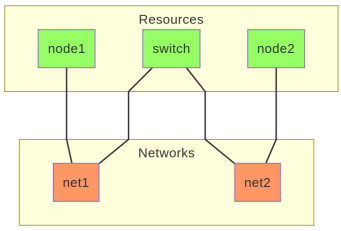

Fig. 5: Example virtual topology represented as a bipartite graph: user-controllable resources are shown in green, while networks are shown in red.

A resource can be described through constraints: at least 10 CPU cores, at least 2 Ethernet 10Gbit/s network interfaces… Alternatively, the resource description can directly refer to specific hardware resources from a testbed (e.g. using a cluster name).

In the graph, a node of class *network* can represent many different kinds of real networks. The simplest network is a L1 point-to-point connection, i.e. a simple cable between two devices. In that case, only two *resources* can attach to this *network* in the graph. More complex networks include L2 networks (e.g. provided by a VLAN by the testbed), or the concept of "LAN" from Emulab, defined as "a clique of nodes [resources] with full bisection bandwidth" [19].

The next step is to be able to describe this graph programmatically. Several existing network testbeds provide a way to describe a topology in a high-level language such as Python. We implement the example topology from Fig. 5 in two testbeds: Emulab and Grid'5000. For the sake of brevity, we only focus on the topology aspect and omit other configuration options (e.g. which operating system should be deployed on which resource).

Code 1 example is written in Python using `geni-lib`, simplified from the Emulab documentation [20], with two servers and a **Dell S4048** network switch:

```
# Add a raw PC to the request and add an interface
node1 = request.RawPC("node1")
iface1 = node1.addInterface()

# Add the Switch and give it some interfaces
mysw = request.Switch("mysw");
mysw.hardware_type = "dell-s4048"
swiface1 = mysw.addInterface()
swiface2 = mysw.addInterface()
node2 = request.RawPC("node2")
iface2 = node2.addInterface()

# Add L1 link from node1 to mysw
link1 = request.L1Link("link1")
link1.addInterface(iface1)
link1.addInterface(swiface1)

link2 = request.L1Link("link2")
link2.addInterface(iface2)
link2.addInterface(swiface2)
```

Code 1: Example topology from Fig. 5 defined in Python with `geni-lib`, targeted at Emulab

Code 2 is using EnOSlib [21] on Grid'5000, with `fleckenstein` referring to classical x86 servers and `engelbourg` referring to a **Edge-core Wedge100BF** network switch:

```
net_conf = dict(type="kavlan", site="strasbourg")
net1 = en.G5kNetworkConf(**net_conf)
net2 = en.G5kNetworkConf(**net_conf)

conf = (
  en.G5kConf()
  .add_network_conf(net1)
  .add_network_conf(net2)
  .add_machine(
    cluster="fleckenstein",
    nodes=1,
    secondary_networks=[net1]
  )
  .add_machine(
    cluster="fleckenstein",
    nodes=1,
    secondary_networks=[net2]
  )
  .add_machine(
    cluster="engelbourg",
    nodes=1,
    secondary_networks=[net1, net2]
  )
)

provider = en.G5k(conf)
provider.init()
```

Code 2: Example topology from Fig. 5 defined in Python with EnOSlib [21], targeted at Grid'5000

Other techniques can be used to describe the topology of an experiment: Heat templates on Chameleon Cloud [22], and the Fablib Python library [23] on the FABRIC testbed [24].

While these approaches have differences, they all adopt a declarative structure to describe the resources, the networks, and their connections. We believe that a declarative approach has good properties: it allows to validate correctness properties on the whole topology before actually allocating resources on the testbed, and it is also a key requirement to be able to choose the appropriate physical resources that fit the desired topology.

b) *Configuring physical resources to setup the topology:*

Setting up the virtual topology usually boils down to two categories of actions: **1) allocate and configure physical nodes**, e.g. reboot them, deploy an OS on them, configure their network interfaces, and allow the SSH key of the experimenter; **2) allocate and deploy the desired networks and network connections**, e.g. by deploying VLANs on the infrastructure and configuring the physical ports on which the nodes are connected. We only focus on this second category.

Existing testbeds use dedicated tools to reconfigure the network dynamically: KaVLAN on Grid'5000; `snmpit` [25] on Emulab; Neutron and Networking-Generic-Switch on Chameleon Cloud; and a Cisco Network Services Orchestration (NSO) controller on FABRIC. These reconfiguration tools all expose a high-level interface (often called the northbound API) through which network configuration requests can be made. Then, each tool exploits a low-level interface (often called the southbound API) to communicate with the actual network devices and update their configuration. The southbound API can either use SNMP, SSH, or the NETCONF protocol with appropriate YANG models.

Another interesting aspect of YANG models is the possibility to use them as intermediate representation of network topologies. Such a representation has the potential to be agnostic to the testbed and to the specific network hardware to be configured. A potentially useful model to achieve this is the YANG Data Model for Network Topologies from RFC 8345. However, to our knowledge, this intermediate representation approach has not been explored in current testbeds: YANG and NETCONF are only used as the low-level southbound API by the Cisco NSO controller on FABRIC, likely using Cisco-specific YANG models.

C. Network Operating System deployment

Changing the network operating system by the network while using this same network can be tough. It can be compared to sawing off the branch you are sitting on. Fortunately, innovative solutions have been created such as Open Network Install Environment (ONIE) [26], that inherit from ONL [27]. ONIE is a minimal operating system that its only functionality is to connect to a network and waiting to receive the image to install. Once an image is installed, the primary boot partition is automatically set to the installed OS one. If the operating system gets uninstalled, then the primary boot partition is set back to ONIE's. One of the two limitations regarding ONIE is that it has to come pre-installed with non-white box or bare metal switches. The other main limitation is that images that ONIE can install have to be tweaked before proceeding, thus limiting the available options .

Various other operating system deployment tools are available, including Kadeploy, which is actively being developed within the Grid'5000 testbed. Kadeploy is a scalable and efficient deployment system designed to facilitate the installation of operating systems across multiple nodes simultaneously. It provides experimenters with the ability to customize post-installation configurations, ensuring a tailored deployment process. While initially developed for general-purpose computing clusters, its flexibility allows for adaptation to network environments, including white-box switches and ONIE-compatible images. However, its integration with network operating systems remains a work in progress, requiring further development to fully support the specific needs of network infrastructure experimentation.

D. Service Orchestration

Service orchestration plays a crucial role in the efficient deployment and management of experiments within our testbed. The flexibility of our infrastructure allows experimenters to deploy off-the-shelf orchestrators as part of their research, enabling the evaluation of orchestration frameworks in real-world scenarios. This capability is particularly relevant for studying performance, scalability, and resilience aspects of different orchestration solutions in a controlled yet realistic environment.

To facilitate experiment execution, we provide support for various orchestration tools that experimenters can leverage to manage their deployments. These tools include widely used configuration management and automation frameworks such as Ansible, as well as more specialized solutions like EnOSlib [21] which offer tailored functionalities for network and distributed system experiments. By integrating these tools, experimenters can automate system configuration, deploy network services, and manage complex experimental workflows with minimal overhead. This approach ensures reproducibility, scalability, and efficiency within the testbed.

Regarding the Computing Continuum, service orchestration will play a significant role in future experiments. Kubernetes remains the most widely used solution for simulating or emulating a continuum-like computing model, as few alternatives have emerged over time. However, in line with the European initiative Future European Platforms for the Edge: Meta-Operating Systems—part of the Horizon Europe program —new Meta Operating Systems (Meta OS) have started to appear. Each proposed solution offers a unique perspective on achieving the Computing Continuum, including NEMO [28], FLUIDOS [29], ICOS [30], NebulOuS [31], AerOS [32], and Nephele [33]. These Meta Operating Systems are not all intended to be deployed on the same computing tiers. For instance, Nephele focuses exclusively on edge devices. Whereas, FLUIDOS lays from cloud to edge.

The Computing Continuum experiment framework, E2Clab [34], built on top of EnOSlib and targeted at Grid'5000, was introduced in 2020. Its adoption facilitates real-world testing and benchmarking of Meta OS service orchestration. Given that both E2Clab and our proposed testbed are based on Grid'5000, we believe they are compatible and complementary.

V. Discussion & future work

In this work, we introduced a conceptual testbed designed to enhance the flexibility, scalability, and replicability of Computing Continuum experiments. We outlined three distinct usage scenarios, each addressing different networking challenges and research needs. Additionally, we reviewed existing network operating systems and research infrastructures, highlighting their limitations and compatibility with our approach. By enabling dynamic OS replacement and adaptable network topologies, our testbed aims to bridge the gap between theoretical advancements and practical deployments in next-generation networking environments. While our testbed is conceptual, we based our work on existing research infrastructures such as Grid'5000 to envisage a realistic implementation. Practical validation still remains an open challenge.

A limitation when deploying network operating systems is the necessity to use ONIE-compatible OS images, thus limiting the available options and slowing down OS development process. This limitation is peering up with the non-availability of Meta OS images for the time being. A future work for the Meta OS research field could be to get a step closer to NOS installation solutions such as ONIE. To follow with service orchestration, a Computing Continuum experimentation framework like E2Clab can be pushed onto our testbed to shape a powerful and reliable testing workflow.

The most ambitious future work for Computing Continuum experimentation at large scale is to enable interconnectivity and compliance of various research infrastructures. Such a research advancement merged with our testbed proposal would empower the test of life-sized complex experiments. Finally, establishing collaborations with industry and research initiatives will be crucial to ensuring real-world applicability and aligning with evolving networking and computing standards.

References

[1] S. Dustdar, V. C. Pujol, and P. K. Donta, "On Distributed Computing Continuum Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4092–4105, Apr. 2023, doi: 10.1109/TKDE.2022.3142856.

[2] M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, "Computing Without Borders: The Way Towards Liquid Computing," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2820–2838, Jul. 2023, doi: 10.1109/TCC.2022.3229163.

[3] J. Marino and F. Risso, "Is the Computing Continuum Already Here?," no. arXiv:2309.09822. arXiv, Sep. 2023. doi: 10.48550/arXiv.2309.09822.

[4] S. Fdida *et al.*, "SLICES, a Scientific Instrument for the Networking Community," *Computer Communications*, vol. 193, pp. 189–203, Sep. 2022, doi: 10.1016/j.comcom.2022.07.019.

[5] T. Friedman, R. McGeer, B. C. Senel, M. Hemmings, and G. Ricart, "The EdgeNet System," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, Oct. 2019, pp. 1–2. doi: 10.1109/ICNP.2019.8888122.

[6] B. White *et al.*, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 255–270, Dec. 2003, doi: 10.1145/844128.844152.

[7] J. Mambretti, J. Chen, and F. Yeh, "Next Generation Clouds, the Chameleon Cloud Testbed, and Software Defined Networking (SDN)," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, Singapore, Singapore: IEEE, Oct. 2015, pp. 73–79. doi: 10.1109/ICCCRI.2015.10.

[8] R. Bolze *et al.*, "Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed," *The International Journal of High Performance Computing Applications*, vol. 20, no. 4, pp. 481–494, Nov. 2006, doi: 10.1177/1094342006070078.

[9] "FogGuru Project – Training the Next Generation of European Fog Computing Experts." [Online]. Available: http://www.fogguru.eu/

[10] Frédéric Le Mouël, "YOUPI: Feedbacks, Ongoing and Future Works of a Fog/ Edge Computing Platform."

[11] A. Massouri *et al.*, "CorteXlab: An Open FPGA-based Facility for Testing SDR & Cognitive Radio Networks in a Reproducible Environment," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2014, pp. 103–104. doi: 10.1109/INFCOMW.2014.6849176.

[12] "Grid'5000 Site of Strasbourg and Toulouse." Accessed: Feb. 13, 2025. [Online]. Available: https://www.grid5000.fr/Core2EdgeNetworkXP/

[13] "KaVLAN - Grid5000." Accessed: Jan. 29, 2025. [Online]. Available: https://www.grid5000.fr/w/KaVLAN

[14] C. Wang, Z. Li, X. Huang, and P. Zhang, "Inferring the Average as Path Length of the Internet," in *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Sep. 2016, pp. 391–395. doi: 10.1109/ICNIDC.2016.7974603.

[15] P. Sebos, J. Yates, G. Li, D. Rubenstein, and M. Lazer, "An Integrated IP/Optical Approach for Efficient Access Router Failure Recovery," in *Optical Fiber Communication Conference, 2004. OFC 2004*, p. 289–. [Online]. Available: https://ieeexplore.ieee.org/document/1359261

[16] "Assembling VoLTE CDRs Based on Network Monitoring – Challenges with Fragmented Information," in *ResearchGate*, doi: 10.23919/INM.2017.7987410.

[17] C. Adjih *et al.*, "FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, Italy: IEEE, Dec. 2015, pp. 459–464. doi: 10.1109/WF-IoT.2015.7389098.

[18] C. V. Dadauto, N. L. S. da Fonseca, and R. d. S. Torres, "Data-Driven Intra-Autonomous Systems Graph Generator." Accessed: Jan. 28, 2025. [Online]. Available: http://arxiv.org/abs/2308.05254

[19] F. Hermenier and R. Robert, "How To Build a Better Testbed: Lessons From a Decade of Network Experiments on Emulab," 2012, p. 1. [Online]. Available: https://hal.science/hal-00710449

[20] Eric Eide *et al.*, "The Emulab Manual." Accessed: Feb. 17, 2025. [Online]. Available: https://docs.emulab.net/advanced-topics.html#%28part._user-controlled-switches%29

[21] R.-A. Cherrueau *et al.*, "EnosLib: A Library for Experiment-Driven Research in Distributed Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1464–1477, Jun. 2022, doi: 10.1109/TPDS.2021.3111159.

[22] Chameleon Cloud, "Complex appliances — Chameleon Cloud Documentation." Accessed: Feb. 18, 2025. [Online]. Available: https://chameleoncloud.readthedocs.io/en/latest/technical/complex.html#heat-orchestration-templates

[23] Paul Ruth and Komal Thareja, "FABRIC testbed — fablib documentation — node." Accessed: Feb. 18, 2025. [Online]. Available: https://fabric-fablib.readthedocs.io/en/latest/node.html

[24] I. Baldin *et al.*, "FABRIC: A National-Scale Programmable Experimental Network Infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, Nov. 2019, doi: 10.1109/MIC.2019.2958545.

[25] Emulab, "tbsetup/snmpit · master · emulab / emulab-devel · GitLab." Accessed: Feb. 18, 2025. [Online]. Available: https://gitlab.flux.utah.edu/emulab/emulab-devel/-/tree/master/tbsetup/snmpit

[26] "Open Network Install Environment — Open Network Install Environment Documentation." Accessed: Jan. 10, 2025. [Online]. Available: https://opencomputeproject.github.io/onie/

[27] "ONL Home." [Online]. Available: http://opennetlinux.org/

[28] "HOME - NEMO META-OS." [Online]. Available: https://meta-os.eu/

[29] "FLUIDOS." [Online]. Available: https://fluidos.eu/

[30] "Project ICOS." [Online]. Available: https://www.icos-project.eu/

[31] "Home - NebulOuS." [Online]. Available: https://nebulouscloud.eu/

[32] "aerOS Project." [Online]. Available: https://aeros-project.eu/

[33] "Home | Nephele." [Online]. Available: https://nephele-project.eu/

[34] D. Rosendo, P. Silva, M. Simonin, A. Costan, and G. Antoniu, "E2Clab: Exploring the Computing Continuum through Repeatable, Replicable and Reproducible Edge-to-Cloud Experiments," in *Cluster 2020 - IEEE International Conference on Cluster Computing*, Kobe, Japan, Sep. 2020, pp. 1–11. doi: 10.1109/CLUSTER49012.2020.00028.