Multi-Cluster Orchestration of 5G Experimental Deployments in Kubernetes over High-Speed Fabric

Ilias Syrigos*, Nikos Makris*†, and Thanasis Korakis*†
*Dept. of Electrical and Computer Engineering, University of Thessaly, Greece
†Centre for Research and Technology Hellas, CERTH, Greece
Email: ilsirigo@uth.gr, nimakris@uth.gr, korakis@uth.gr

***Abstract*—Network Functions Virtualization has been a key enabler for the wide adoption of cloud-native functions for workloads. Well-established orchestration frameworks, such as Kubernetes, optimize network operation to meet the networking requirements of deployed workloads, while providing a flexible API for fine-grained control throughout the lifecycle of the workload. As a result, these tools can be used effectively to provide access to distributed 5G experimental facilities, even across continents. However, resource clusters may belong to distinct administrative authorities; therefore, cluster integration must occur by exposing only the necessary information and services. In this paper, we suggest employing SUSE Rancher for managing multi-cluster Kubernetes deployments and utilize the Submariner framework in order to securely export services and establish connectivity across clusters. We evaluate the efficacy of such integrated framework and its various configurations over a high-speed networking fabric (up to 25 Gbps) connecting the various clusters and enabling 5G and beyond experimentation.**

***Index Terms*—Kubernetes, multi-cluster, orchestration, 5G, testbeds**

I. Introduction

Fifth-generation networks (5G) have completely transformed our digital world interactions and communications. They are the primary enablers for applications requiring higher data rates and ultra-low latency, including autonomous vehicles, augmented reality, and smart cities. 5G represents a paradigm shift from traditional wireless networking by softwarizing and virtualizing network functions (SDN/NFV) and multiplexing them with a cloud-native approach of containerized microservices to deliver fine-grained control and management. This approach enables network operators to reduce operational costs, enhance system flexibility and scalability, and simultaneously enable the development and provisioning of services and applications with variable throughput and latency requirements. Utilizing softwarization and virtualization/containerization, the distributed deployment of a 5G network is now possible, where the cloud-native 5G Core Network can be instantiated in a cloud environment with abundant resources, while the 5G RAN can be deployed at the network's edge to enable real-time, low-latency applications.

Parallel research initiatives from standardization consortia, organizations, industry, and academia have begun to address the limitations of 5G and evolve it into the 6th generation (6G). To this end, research infrastructures and testbeds play an important role by providing experimental platforms that enable realistic evaluation of concepts, protocols, and services to researchers from academia and industry. In addition, they offer a controlled environment in which researchers can manage the deployment of their experiments and fine-tune their parameters in a standardized, well-defined, and softwarized manner, thereby ensuring reproducibility.

To facilitate truly large-scale, realistic experimentation, however, researchers require a variety of frequently highly heterogeneous high-capacity resources. The availability of such resources in research infrastructures exists on a global scale, but the problem is that these infrastructures are more isolated silos of resources than a unified research space. This results in a fragmented view of the total available resources and hinders the deployment of fully distributed large-scale applications.

To address this challenge, it is necessary to integrate the various research facilities into a single one that provides an environment for the seamless and simple access to heterogeneous experimental resources from various sites and locations around the globe. In such a situation, a central hub is required for the management and orchestration of resources and experiments, while direct access to individual sites for the same operations is also required.

For such an integration, virtualization and containerization of network functions (Virtual/Container Network Functions) and services will be essential to realizing its potential, because they will decouple the network from its physical infrastructure and heterogeneity. Additionally, the adoption of orchestration frameworks such as Kubernetes [1] is essential for assisting in bridging the gaps in integration between the various research sites.

In this paper, we design and implement a fully-fledged multi-domain orchestration framework for providing centralized orchestration and fine-grained control of CNFs, VNFs and services over Kubernetes-based resource clusters, thereby facilitating the discovery and interconnection of services existing in distinct clusters. We employ an experimentally driven approach in order to determine the overhead and tradeoffs of the adopted solution, using isolated clusters in the NITOS testbed, the Greek node of the SLICES-RI platform [2]. We assess this setup by quantifying the performance impact of such a distributed deployment and discuss the different

The research leading to these results has received funding from the European Horizon 2020 Programme for research, technological development and demonstration under Grant Agreement Number No 101008468 (H2020 SLICES-SC). The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only.

2023 IEEE Globecom Workshops (GC Wkshps) | 979-8-3503-7021-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/GCWKSHPS58843.2023.10465054

deployment options and limitations that they present.

The remaining sections of this paper are structured as follows: The section II provides a summary of relevant concepts and background information. Section III describes in detail each of our contributions while presenting the overall architecture of our developed framework. In section IV, a proof-of-concept use case deployed on multiple clusters is used to evaluate the framework. Section V, concludes the paper by reviewing our proposed scheme and findings and identifying future research directions.

II. Background and Motivation

In this section, we present some background concepts regarding virtualization and containerization, as well as their application in the cloud-native approach of the 5G Service-Based Architecture (SBA), where control plane functionality is provided by an interconnected chain of Network Functions (NFs).

A. Virtualization and Containerization

By enabling the creation of virtual instances of hardware platforms, storage, and network resources, virtualization enables the decoupling of physical resources from the underlying hardware. It enables the concurrent execution of multiple such instances, usually virtual machines (VMs), on commercially available hardware (server, server cluster), while providing isolation and security to software operating on them. In addition, it offers cloud providers flexibility and considerably improves the utilization of their hardware resources by enabling the optimal placement and migration of virtual machines, which contributes to meeting SLA requirements. Another advantage of virtualization is the increased availability of services, as hardware failures can be quickly recovered by migrating or replicating VMs to another physical machine. All of the aforementioned advantages have resulted in a significant reduction of CAPEX and OPEX for infrastructure providers, and have played a significant role in the development of cloud technologies that provide on-demand services and applications by sharing a pool of computing, storage, and network resources.

Containerization is a well-established technology that service providers and developers are swiftly adopting. Containerization is fundamentally a lightweight form of virtualization, wherein a container is a self-contained software package with its dependencies. The major difference between a container and a virtual machine lies at the level of abstraction. Containers are software abstractions of the operating system, whereas virtual machines are software abstractions of the hardware platform. The kernel space of the operating system is shared by containers running on the same host machine, despite the fact that their processes are entirely isolated and protected. Containerization is gaining popularity because it offers a significant advantage over virtual machines, namely that containers are more lightweight and have less overhead than VMs, which include a complete operating system stack and are consequently much slower to boot. This provides enhanced portability and scalability, efficient load balancing, rapid deployment, and overall improved resource utilization and efficiency. For all these benefits, the NFV industry and mobile operators are starting to favor container-based platforms and Containerized Network Functions (CNFs) over the conventional VM-based NFV platforms and the traditional Virtualized Network Functions (VNFs).

B. Network Disaggregation

At the same time, 5G networks have introduced a number of innovations in terms of network deployment and overall operator flexibility, enabled by the adoption of the concept of disaggregation. 5G introduces two forms of disaggregation by implementing: 1) the functional split of the base station units to simpler elements (e.g. CU/DU/RU split), with the CU/DU components being able to be executed at the cloud/edge; and 2) the Control/User-plane disagreggation across the different elements at the stack (e.g. CU disaggregation to CU-U and CU-P communicating over the E1 interface, or the O-RAN interfaces for programming the network using xApps). Through the adoption of Service-Based Architecture and the execution of discrete network functions, 5G Core Network components are also being disaggregated. As a result, cellular network operators can greatly benefit from the virtualization of their components, which enables their network to be executed in a cloud-native manner, thereby reducing their capital and operational costs and making the deployment of the network more flexible.

C. Workload Orchestration

The Management and Orchestration (MANO) of these virtualized functions is of utmost importance to the network operator, as it ensures the proper chaining of services, the establishment of appropriate datapaths among them, and their seamless operation during their lifecycle. There are several options for the orchestration of services, depending on their operation and the environment for which they are intended. The Kubernetes (K8s) ecosystem is one of the most prominent frameworks for orchestration. K8s automates the instantiation, scaling, and management of container-based applications and microservices, while specialized plugins facilitate the network establishment and service exposure within the cluster. K8s claims to be one of the most successful open-source orchestration platforms compatible with all modern container technologies at present. K8s provides integrated monitoring functionalities for monitoring and restoring application health, efficient resource allocation and storage organization, automated load-balancing and replication of high-traffic containers, management of application runtime state and control of deployments and updates, enabling the efficient use of physical resources [3], [4].

Although K8s provides a well-structured API for deploying workloads, it can only manage operations within a single cluster. There is no organized off-the-shelf solution for integrating other clusters (possibly under different administration domains) into a single centrally managed facility. In this work, we progress beyond these limitations and employ a fully-fledged solution that enables the cross-domain orchestration of several K8s clusters that are administered by distinct

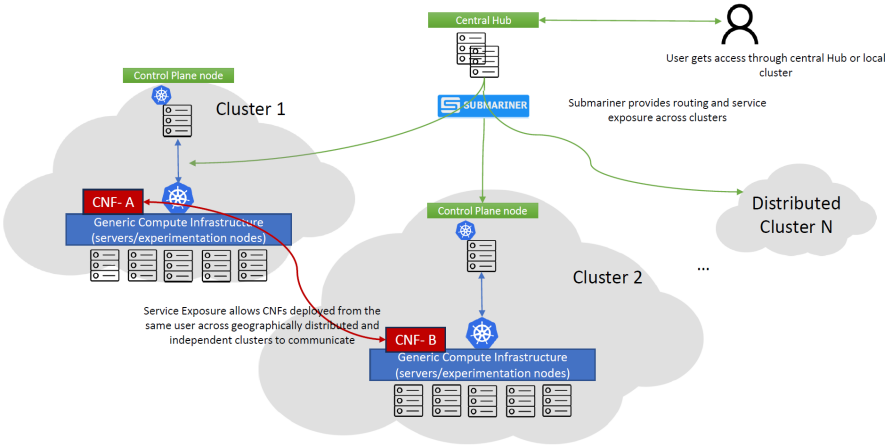


Fig. 1: The overall under-study architecture: we consider geographically distributed clusters that operate independently. On top, the framework adds a management layer through a central hub, and exposes network functions and services to other clusters.

administration authorities. We develop the functionality for securely exposing services across integrated clusters and conduct experiments with workloads across the integrated resource continuum. In the following section, we describe the overall architecture and components that enable this functionality.

III. Implementation Design

In this section, we provide the design and implementation of our framework that facilitates the deployment and orchestration of the 5G Core Network and the 5G RAN across multiple distributed research infrastructures (multiple domains) by exposing CNFs and providing transparent inter-cluster connectivity.

A. Overall Architecture

Figure 1 is a visual representation of the overall architecture under consideration. It is comprised of several independent research infrastructures, displayed as nodes, each of which represents a resource island. A container management system based on Kubernetes manages each cluster's of resources directly. On top of the clusters a central hub exists, where a multi-domain orchestrator is deployed and directly interfaced with the individual Kubernetes-based clusters, enabling the deployment and management of containerized workloads across domains. The experimenters can login in from the central hub to a graphical user interface that allows them to inspect clusters and their worker nodes, deploy and configure pods and deployments, monitor the status of infrastructure and workloads, and define security and network policies. However, the configuration is flexible in the sense that the experimenters can potentially log in directly to the cluster management portal of an individual node with the same credentials and deploy their experimental workload locally.

Researchers can leverage such an architecture in order to experiment with existing and develop new concepts on 5G networks disaggregation and slicing in a large-scale multi-domain environment. Open-source implementations assist them, with the OpenAirInterface5G platform (OAI) [5] for the 5G RAN and Core components, srsRAN for the RAN [6], and Open5GS and free5GC [7] for the Core being the most well-known frameworks. Although all implementations support the majority of functionalities, OAI has a larger user base and provides additional features, such as disaggregation of RAN and support for multiple SDR devices.

B. Multi-domain Orchestrator

The European Telecommunications Standards Institute's (ETSI) NFV architectural framework forms the basis for NFV specification and management in 5G networks. Specifically, the ETSI NFV MANO specification [8] defines the framework under which VNFs are provisioned, deployed, configured, and managed throughout their life-cycle. This framework is comprised of three major functional blocks: a) the NFV Orchestrator (NFVO), b) the VNF Manager (VNFM), and c) the Virtualized Infrastructure Manager (VIM). Initially, VNF management was primarily focused on virtual machine handling, but CNF support has since been introduced. Several compliant frameworks, such as OSM and ONAP [9], have been developed to implement the NFV MANO architecture, but a cloud-native approach is required to offer multi-domain orchestration support over existing containerized infrastructures and provide a flexible and simple method for managing multi-site experimentation.

To this end, we propose employing SUSE Rancher as the orchestrating framework for VNFs and CNFs. Rancher is a Kubernetes management tool that enables the deployment and management of clusters on any provider or location, as well as allowing the import of existing, distributed Kubernetes clusters that operate in different regions, cloud providers, or on-premises. It supports multiple flavors of Kubernetes, including K8s, K3s, and RKE/RKE2, and one of its main advantages is its centralized authentication and role-based access control (RBAC), which enables administrators to manage multiple clusters and their members from a central location. Though it is primarily targeting the deployment of CNFs, VNF deployment is also possible through extensions such as

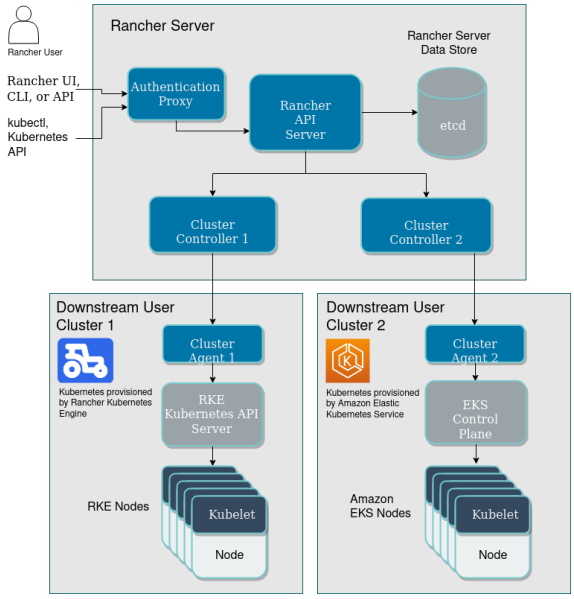


Fig. 2: Rancher architecture [11].

KubeVirt [10]. Deploying Rancher on a central hub to manage multiple Kubernetes-based clusters across distributed regions enables access to these clusters, management of their members and nodes, management of their persistent volumes and storage classes, management of their projects, namespaces, and workloads, and deployment and configuration of monitoring, logging, and alerting tools.

Figure 2 illustrates the high-level architecture of Rancher, in which the majority of Rancher's software components reside on the Rancher server. These components coordinate to provide access to multiple downstream clusters. Before initiating API calls to the master of the downstream Kubernetes cluster, a Rancher user must first authenticate with the Rancher server, and specifically with the Authentication Proxy, which adds the proper Kubernetes impersonation payload. The communication between Rancher and Kubernetes clusters is performed using a service account that identifies pod-running processes. A Cluster Agent is deployed on each downstream cluster to establish a tunnel with the dedicated Cluster Controller on the Rancher server. The Cluster Agent connects to the Kubernetes API of its cluster, manages workloads and pods, implements global policy roles and bindings, and communicates with the cluster controller regarding health status, events, and statistics. The Cluster Controller is responsible for configuring access control policies, monitoring events and resource changes, and provisioning the downstream cluster.

Overall, we believe that Rancher is the best option for managing containerized infrastructures with multiple clusters. In comparison to other multi-cluster alternatives [12], such as Kubefed [13], it offers advantages such as centralized authentication and RBAC for downstream clusters, as well as a superior graphical interface that facilitates user-friendliness. Compared to a single cluster Kubernetes deployment consisting of a master node on the central hub and worker nodes located at the various sites, it provides autonomy at each site,

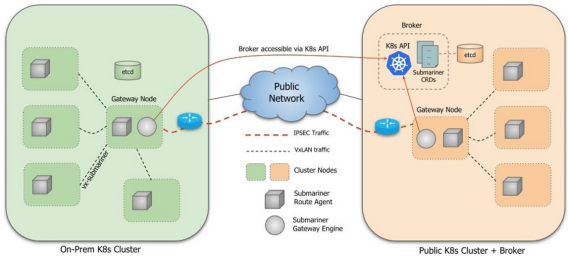


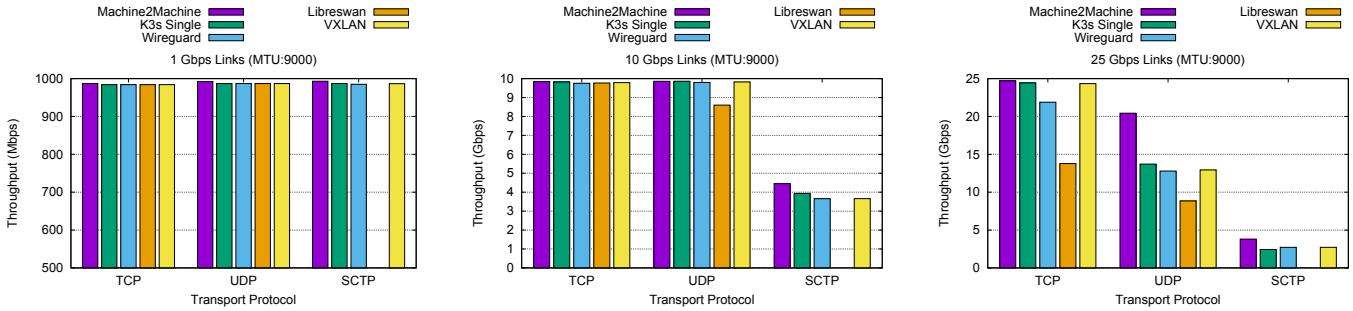
Fig. 3: Submariner network stitching among clusters [14].

ensuring availability even when the central hub is unavailable.

C. Multi-domain Network Stitching

Even though the multi-domain orchestrator provides the ability to manage multiple containerized clusters deployed on different sites, it is not sufficient to deploy a workflow, or in our case, a 5G experimental network, across these clusters. The missing piece is network stitching that given multiple network subnets as input, will produce a single end-to-end network as output. In this paper, we propose that the Submariner framework (Fig. 3) [14] assumes the role of network stitching module. Submariner is a sandbox project of the Cloud Native Computing Foundation (CNCF) that offers encrypted or unencrypted L3 cross-cluster connectivity, service exporting and discovery across clusters, and support for the interconnection of clusters with overlapping CIDRs.

Submariner consists of multiple components to ensure connectivity between clusters. The first of these components is the Gateway Engine, which is deployed in each cluster and which primary function is to create tunnels to all other clusters. The Gateway Engine supports three tunneling implementations: a) Libreswan [15], an IPsec VPN protocol; b) WireGuard [16], an additional VPN tunneling protocol; and c) unencrypted tunnels via VXLAN. The second component is the Route Agent, which is deployed on each node of each cluster and is responsible for establishing VXLAN tunnels and routing cross-cluster traffic from the node on which it resides to the node that hosts the Gateway Engine, which will then forward the traffic to the destination cluster. The Broker is a singleton component of Submariner that is deployed on a cluster whose Kubernetes API is accessible by all clusters; therefore, the central hub node is its ideal location for the architecture under consideration. Broker enables the Gateway Engines to discover one another and stores metadata such as the Service and Pod CIDRs for each cluster. It is a central component that enables connectivity between clusters, yet its availability is not always required because Gateway Engines and Route Agents will continue to route traffic based on the most recent information even if it is unavailable. Lighthouse Agent and Lighthouse DNS Server facilitate Service Discovery for Submariner. The Lighthouse Agent is deployed in each cluster and, in coordination with the Broker, imports every service exported by the other clusters. These imports are utilized by the Lighthouse DNS Server to resolve DNS requests received from the Kubernetes cluster's configured CoreDNS service. In the event that a service is deployed in



(a) Throughput results for 1 Gbps links. (b) Throughput results for 10 Gbps links. (c) Throughput results for 25 Gbps links.

Fig. 4: Throughput performance results under different settings.

multiple clusters, the Lighthouse DNS server prioritizes the local cluster before selecting a remote cluster in a round-robin fashion. Eventually, an important Submariner component is the Globalnet Controller, which supports overlapping CIDRs in clusters by establishing a virtual network with a global CIDR.

IV. Evaluation

In this section, we discuss the evaluation results of our proposed framework, which provides multi-domain connectivity and orchestration and enables transparent deployment of 5G experimental networks across distributed testbeds.

To evaluate our approach, we utilize the NITOS testbed [17]. We employ two single-node clusters based on K3s that are incorporated into a single Rancher instance, each of which runs on a separate node. The specifications of each node are identical, as shown in Table I. We measure and compare the throughput and latency (RTT) obtained in the following cases: a) when using directly the bare metal machines (Machine2Machine), b) when both nodes are integrated in a single K3s cluster (K3s single) that uses Flannel as its default Container Network Interface (CNI), or c) when each node is hosting a separate K3s cluster, and the deployed services are exposed using Submariner, configured with the three different tunneling options (Wireguard, Libreswan, and VXLAN). Notably, the K3s clusters are deployed directly on the bare metal servers and not over a virtualized Openstack infrastructure, which is a common deployment method, because enabling VXLAN tunneling over the Openstack provider network would require explicit configuration.

TABLE I: Configurations used in the experiments

Experiment Parameters	Value
Processor	Intel Core i7-11700K @ 3.60GHz
Memory	64GB
NIC (1 Gbps experiments)	Realtek RTL8125 2.5Gbps Ethernet
NIC (10 & 25 Gbps experiments)	Netronome Agilio CX25Gbps SFP28
Operating System	Ubuntu 22.04 server (Linux Kernel v6.4.3)
Rancher Version	v 2.7.3
K3s Version	v1.25.11 +k3s1
Submariner Version	v0.15.2
Traffic Generator	iperf v.3.9

In order to provide a full overview of the performance of the various connectivity options, we obtained the aforementioned metrics for three different link speeds (1 Gbps, 10 Gbps, 25 Gbps) by configuring the NICs appropriately and deactivating the auto-negotiation feature. The MTU of the physical interface was set to 9000 bytes, and since both Flannel and Submariner support auto-MTU configuration, all established tunnels and interfaces were configured to account for the tunneling overhead, with Wireguard being the only exception that required manual configuration. Using iperf3 and the three transport protocols TCP, UDP, and SCTP, throughput was measured, while RTT was determined using ping. We considered evaluating the SCTP protocol, as it is used for signaling traffic transfer in 5G and beyond 5G networks. Similarly, we evaluated the network's latency, as it plays a crucial role in serving uRLLC traffic across clusters. All presented results display the average throughput and RTT measurements as the average of 50 consecutive trials.

Figure 4 illustrates our results regarding the attained throughput. When using bare metal servers or the Flannel CNI in a single cluster configuration, the achieved TCP throughput is very close to the maximum allowable transfer rate across all experiment scenarios (1 Gbps, 10 Gbps, and 25 Gbps). Comparing the Submariner tunneling options, we find that the VXLAN cable driver is capable of achieving maximum speed, which is close to the bare metal transferable rate. Wireguard comes close to that value, being capable of achieving a throughput of 22 Gbps. Libreswan is the cable driver with the lowest performance, achieving a maximum of 12 Gbps in the 25 Gbps experiments. In both 1 Gbps and 10 Gbps links, all protocols reach their maximal throughput.

In the case of UDP transfers, where by default 4 KByte buffer size is used, bare metal and single cluster can attain speeds close to the 10 Gbps maximum rate. However, as the underlying capacity is increased to 25 Gbps, UDP transfer endures high losses, resulting in a maximum of 21 Gbps for the bare metal case and 14.5 Gbps for the single cluster case. Notably, in such high-speed networks, unlike UDP, TCP benefits from the implementations of TCP segmentation and reassembly offloading directly on the card, allowing it to perform better in comparison. VXLAN outperforms the other options in the evaluation of Submariner cable drivers, achieving throughput close to the single cluster case. Wireguard approaches the performance of VXLAN, obtaining a maximum transferable rate of up to 12 Gbps, while Libreswan is the solution with the worst performance, achieving a maximum transferable rate of 9 Gbps.

As our final transport protocol, SCTP with a window size

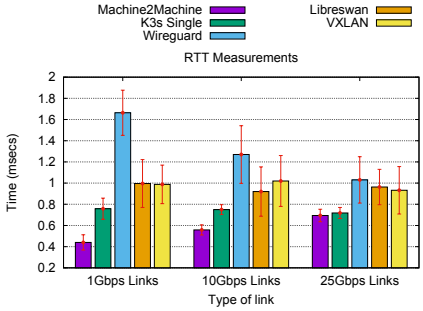


Fig. 5: RTT measurements for the different cable drivers.

of 4 MBytes is utilized. SCTP achieves a maximum transfer rate of 4.5 Gbps for bare metal transfers, considerably lower than TCP and UDP. When using a single stream for sending/receiving, these rates reflect adjustments to the operation of SCTP, increasing parameters such as the segment size and the maximum surge traffic over the standard implementation. SCTP appears to adapt transfer windows in a less reactive manner than TCP, resulting in a markedly lower throughput. Regarding the various Submariner tunneling options, the performance of VXLAN and Wireguard are comparable. In 25 Gbps links, both cable drivers obtain lower throughput than in 10 Gbps links, 2.7 Gbps and 3.6 Gbps, respectively, a noteworthy observation. Regarding Libreswan, we were unable to transmit SCTP-based traffic because support does not appear to be enabled by default; consequently, the presented results do not indicate any packet transmission.

Figure 5 illustrates the average RTT observed across all configurations. Similar to the throughput results, the choice of cable driver affects the overall measured RTT, with Wireguard being the worst performing solution, as indicated by the highest deviation from the average values.

In the overall comparison of the different cable drivers for Submariner, VXLAN and Wireguard seem to be notably outperforming Libreswan. Nevertheless, VXLAN does not implement any encryption on top of the transferable traffic, contrary to the other two solutions. The outcomes are consistent with related benchmarking results [18], [19].

Additionally, the transmission of VXLAN traffic needs to be enabled for cloud-based provisioning based on Openstack or equivalent solutions. The two other cable drivers do not require such configuration because they are transferred over UDP in the case of Wireguard and TCP in the case of Libreswan. As a result, they can be simply implemented on any cloud infrastructure without requiring further modifications. In the case that SCTP traffic needs to be exchanged between deployed workloads (e.g. for the case of control traffic for the 5G/beyond 5G network), Libreswan fails to provide such functionality. In overall, Wireguard appears to be the optimal solution in terms of features and performance, attaining high throughput over encrypted and secure tunnels.

V. Conclusion

In this work, we employ an experimentally-driven approach to evaluate Kubernetes-based multi-cluster integration and secure service exposure across different integrated clusters. The solution can be applied as-is for the integration of Kubernetes-based experimental research infrastructures via a central Hub. As indicated by our findings, the selection of the cable driver for the under-study Submariner framework is of the utmost importance, significantly impacting the performance of the workloads deployed on top. Based on the results we obtained, the VXLAN cable driver appears to outperform the other solutions, delivering performance as near as possible to that of the bare metal case. Nonetheless, VXLAN does not provide traffic encryption, hence it is not suitable for use in production-grade environments. Therefore, the next best option is to use the Wireguard driver, which offers encrypted tunnels and comparable performance.

References

[1] M. Luksa, *Kubernetes in action*. Simon and Schuster, 2017.
[2] S. Fdida, N. Makris, T. Korakis, R. Bruno, A. Passarella, P. Andreou, B. Belter, C. Crettaz, W. Dabbous, Y. Demchenko, and R. Knopp, "SLICES, a scientific instrument for the networking community," *Computer Communications*, vol. 193, pp. 189–203, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422002663
[3] I. Syrigos, D. Kefalas, N. Makris, and T. Korakis, "EELAS: Energy Efficient and Latency Aware Scheduling of Cloud-Native ML Workloads," in *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2023.
[4] I. Syrigos, N. Angelopoulos, and T. Korakis, "Optimization of Execution for Machine Learning Applications in the Computing Continuum," in *2022 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2022.
[5] F. Kaltenberger, G. d. Souza, R. Knopp, and H. Wang, "The OpenAirInterface 5G New Radio Implementation: Current Status and Roadmap," in *WSA 2019; 23rd International ITG Workshop on Smart Antennas*, 2019.
[6] srsran-project, "SRS," 2023, [Online], https://github.com/srsran/srsran_project.
[7] F. J. De Souza Neto, E. Amatucci, N. A. Nassif, and P. A. Marques Farias, "Analysis for Comparison of Framework for 5G Core Implementation," in *2021 International Conference on Information Science and Communications Technologies (ICISCT)*, 2021.
[8] M. Ersue, "ETSI NFV management and orchestration-An overview," *Presentation at the IETF*, vol. 88, 2013.
[9] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, "Benchmarking open source NFV MANO systems: OSM and ONAP," *Computer Communications*, vol. 161, pp. 86–98, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420305946
[10] M. Amaral, "KubeVirt scale test by creating 400 VMIs on a single node," in *Free and Open source Software Developers' European Meeting*, 2022.
[11] SUSE, "Rancher," 2023, [Online], https://www.rancher.com/.
[12] L. Osmani, T. Kauppinen, M. Komu, and S. Tarkoma, "Multi-Cloud Connectivity for Kubernetes in 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 42–47, 2021.
[13] F. Faticanti, D. Santoro, S. Cretti, and D. Siracusa, "An Application of Kubernetes Cluster Federation in Fog Computing," in *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2021.
[14] Linux CNCF, "Submariner," 2023, [Online], https://submariner.io/.
[15] Libreswan, "Libreswan - VPN software," [Online], https://libreswan.org.
[16] Wireguard, "Wireguard - Fast, Modern, Secure VPN tunnel," [Online], https://www.wireguard.com/.
[17] N. Makris, C. Zarafetas, S. Kechagias, T. Korakis, I. Seskar, and L. Tassiulas, "Enabling open access to LTE network components; the NITOS testbed paradigm," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015.
[18] P. G. Kannan, B. Salisbury, P. Kodeswaran, and S. Sen, "Benchmarking tunnel and encryption methodologies in cloud environments," *arXiv preprint arXiv:2203.02142*, 2022.
[19] L. Osswald, M. Haeberle, and M. Menth, "Performance comparison of VPN solutions," 2020.