



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna Archivio istituzionale della ricerca

Serverless Computing for QoS-Effective NFV in the Cloud Edge

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Sabbioni, A., Garbugli, A., Foschini, L., Corradi, A., Bellavista, P. (2024). Serverless Computing for QoS-Effective NFV in the Cloud Edge. IEEE COMMUNICATIONS MAGAZINE, 62(4), 40-46 [10.1109/mcom.001.2300182].

Availability:

This version is available at: <https://hdl.handle.net/11585/969575> since: 2024-05-14

Published:

DOI: <http://doi.org/10.1109/mcom.001.2300182>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

Serverless Computing for QoS-effective NFV in the Cloud Edge

Andrea Sabbioni, Andrea Garbugli, Luca Foschini, Antonio Corradi, Paolo Bellavista
 Department of Computer Science and Engineering
 University of Bologna, Bologna, 40136 Italy
 name.surname@unibo.it, andrea.sabbioni5@unibo.it

Abstract—Network Softwarization, particularly Network Function Virtualization (NFV), is revolutionizing networking by separating the hardware from the supported network functions, demonstrating unprecedented flexibility and cost-efficiency. With the rise of Edge Computing, NFV has found a strategic application at the network edge, where network functions are deployed closer to end-users and devices with improved performance and reduced latency. In this context, the Serverless paradigm is gaining attention due to its fine-grained scalability, reduced management efforts, and resource efficiency. Despite the advantages, challenges arise from the ephemeral nature of Serverless functions, leading to latency issues and hampering the creation of efficient coordination and orchestration mechanisms. Adopting this paradigm in resource-constrained scenarios, such as Cloud Edges, is even more challenging due to the insufficient presence of proper Quality-of-Service (QoS) mechanisms in these platforms. This paper introduces our novel Time Effective Middleware for Priority Oriented Serverless Network Function Virtualization (TEMPOS4NFV), a QoS-aware platform specialized for hosting Serverless Network Functions in resource-constrained environments. TEMPOS4NFV addresses resource contention and coordination challenges, offering effective end-to-end QoS differentiation for network workloads executed over multiple federated sites. In addition, this paper examines TEMPOS4NFV hosting a real Virtual Private Network (VPN) case, demonstrating its ability to execute concurrent multi-site QoS differentiated workloads.

Index Terms—Serverless, FaaS, Quality of Service, NFV, Cloud Edge.

I. INTRODUCTION

Network Softwarization, particularly Network Function Virtualization (NFV), has revolutionized traditional networking by transforming network functions into software-based Virtual Network Functions (VNFs). This shift from hardware-centric to software-centric networking offers unprecedented flexibility, scalability, and cost-effectiveness in deploying and managing network services. In this context, Edge Computing emerged as a catalyst opening the distribution of network functionalities closer to end-users and devices, capitalizing on low-latency and high-bandwidth advantages [1].

Along the line of Cloud Services, Infrastructure as a Service (IaaS) and Container as a Service (CaaS) have emerged as reference Cloud Computing models to orchestrate and allocate resources for hosting Virtual Network Functions (VNFs). More recently, the Serverless paradigm, especially in its notable case known as Function as a Service (FaaS), is gaining popularity as a way to orchestrate and host VNFs by taking

advantage of its fine-grained and zero-scaling capabilities, reducing management and development efforts [2].

The capability of Serverless platforms to consume an amount of resources proportional to the current load opens new opportunities for NFV consolidation, by allowing even resource-constrained sites, such as edges or gateways, to host services deployed as VNFs without consuming local resources when they do not execute.

Notwithstanding these advantages, the application of the Serverless paradigm is hampered by major challenges derived from the *ephemerality of functions*. The transient nature of functions generates latency issues during activation, potentially impeding efficient orchestration mechanisms [3]. The absence of *resource retention* from functions can lead to resource contention when activated on request, especially when considering resource-constrained deployment environments. At the same time, the *short liveness of functions* makes it hard to create efficient and articulated coordination mechanisms. Consequently, when considering the Serverless application to NFV, the worst-case executing performance scenario must be considered so reducing the appeal associated with adopting the Serverless paradigm itself.

This paper proposes TEMPOS4NFV, a Quality of Service (QoS) aware, Message Oriented Middleware-aided FaaS platform specialized to host network functions by using FaaS workloads in resource-constrained scenarios. By providing strong QoS differentiation in workflow execution, TEMPOS4NFV aims to support classical NFV solutions, especially in dealing with unpredictable traffic variations. In fact, with its ability to coordinate and compose different prioritization mechanisms, TEMPOS4NFV mitigates the resource contention problems often encountered in other FaaS platforms. Moreover, TEMPOS4NFV further evolves QoS differentiation by proposing a multi-edge federation mechanism able to provide an effective end-to-end QoS differentiation of VNFs composed of functions distributed in separated Edges. Finally, TEMPOS4NFV also integrates specific optimizations that can mutate the behavior of functions by, e.g., dynamically switching from ephemeral to persistent.

To demonstrate TEMPOS4NFV we have validated and evaluated in a real use case scenario of mobile nodes, IoT devices, and a dynamically activated Virtual Private Network (VPN) service. By orchestrating a cross-edge VNF through Serverless TEMPOS4NFV functions, the platform prioritizes critical network traffic and ensures the observability of pro-

cessed packets, exhibiting its novel QoS capabilities. Notably, TEMPOS4NFV represents a pioneering implementation of a FaaS platform that coordinates diverse QoS mechanisms and optimizations to enable cross-edge Serverless VNF execution.

For interested researchers and practitioners, a prototype implementation of TEMPOS4NFV is accessible on GitHub under the repository MMw-Unibo/tempo4nfv.

II. BACKGROUND AND RELATED WORK

Serverless Computing is a novel paradigm promoting a strong absence of developer control on deployment operations, execution environment configuration, and lifecycle aspects of services [2]. The FaaS model, one of the most popular and widespread realizations of Serverless Computing, supports the automatic execution of customer-defined business logic, expressed in the form of a function, executed by triggering events [4]. The association between one or more functions and the particular event is called *function workflow*.

Due to the event-based execution of functions, Serverless Computing provides unprecedented fine-grained scalability and zero-scaling capabilities, with no resource usage when the function does not receive service requests. While this paradigm was originally conceived to exploit unused vast resources of public cloud data centers, its application to edge nodes is raising growing interest for its fast development, its facilitated support to the hosting of new services, and its scalability, thus opening to new perspectives in resource-constrained scenarios [5]. Moreover, the recent advancements in FaaS platforms, including the concept of function composition, have enabled the coordinated execution of multiple functions and the creation of complex workflows by combining simpler functions. In the context of Network Softwarization, the application of the Serverless paradigm primarily allows two approaches. The first approach uses function composition to create a *dynamic and fine-grained scalable control plane* that orchestrates and manages VNFs deployed through IaaS/CaaS models [6]. The second approach leverages fine-grained scalability and fast development to meet the evolving requirements of future networks, by allowing the creation of *VNFs as Serverless functions* [2].

The development of FaaS platforms to host VNFs presents significant challenges in terms of *latency* and *predictability* of the supported workloads [7]. One of the most prominent challenges is a direct consequence of the lifecycle management model of FaaS, which instantiates a new function at each triggering event. This behavior while assuring zero and fine-grained scaling features can result in a non-negligible overhead in re-instantiation and startup of function instances. This aspect, commonly referred to as *Cold Start*, increases response latency and poses a barrier to latency-sensitive applications in QoS-sensitive domains. A significant example along this direction is SERPENS [8] which shows how the improvement of function startup latency efficiency is crucial for the adoption of Serverless NFV.

The Cold Start together with the event-based nature of Serverless Computing also raises unpredictability issues. Workloads that are not easily predictable and/or characterized

by short interleaves, such as those associated with fast-paced periodic events, may not fully benefit from Serverless Computing as it is. Some state-of-the-art Serverless platforms mitigate this problem by introducing a mechanism called *Warm Start*, to enable the usage of an already activated function to process multiple events [9]. However, given the absence of QoS-aware coordination mechanisms in current FaaS supports [3], the Warm Start of a function breaks the Serverless principles by keeping computational resources to that specific workflow.

When dealing with resource-constrained nodes, such as in Edge and on-premise sites [6], the above problems further exacerbate the ephemeral nature of Serverless functions and complicate the coordination of different services in acquiring computing resources. Functions that serve more latency-sensitive workloads may compete for virtualized local resources with other less QoS-sensitive functions running on the same node. These technical challenges are even more prominent when considering the chaining of VNFs, where QoS-sensitive resource management extends across multiple heterogeneous nodes that make up the Serverless support infrastructure [10].

To the best of our knowledge, TEMPOS4NFV is the first FaaS platform offering an efficient QoS differentiation for concurrent VNF instances, by managing multiple functions executing on different sites. By prioritizing access to resources for latency-sensitive VNFs, TEMPOS4NFV effectively mitigates resource contention issues, thereby avoiding the need for over-dimensioning based on worst-case scenarios. As a representative example, this paper presents the notable use case of a Virtual Private Network Function [11] enabling secure and monitored communication among multiple sites. The emergence of IoT devices, characterized by intermittent connectivity and bursts of sudden network traffic, adds a layer of complexity to creating a Virtual Private Network (VPN) in resource-constrained scenarios. Serverless architectures, while advantageous for their scalability and cost-effectiveness, face challenges in suiting the dynamic and sporadic nature of IoT devices. These devices often establish temporary connections, causing unpredictable surges in network traffic that require precise differentiation due to the different criticality levels.

III. THE TEMPOS4NFV ARCHITECTURE

This section introduces TEMPOS4NFV (Time Effective Middleware for Priority Oriented Serverless Network Function Virtualization), a platform designed to effectively manage QoS differentiation in workflow execution, which serves as a valuable solution in supporting classical NFV infrastructure when dealing with variable network flows such as one generated by IoT devices or mobile nodes.

TEMPOS4NFV is based on TEMPOS, a FaaS platform already extensively tested to demonstrate its capabilities in integrating and coordinating different QoS-aware mechanisms to provide an effective QoS differentiation in application layer workflows execution [12]. We present here an evolution of TEMPOS designed to fulfill the specific requirements of Network Softwarization while facilitating the deployment of Network Functions on the TEMPOS4NFV platform, offering

infrastructure providers and developers the dual advantage of seamless, granular scalability in FaaS platforms alongside robust assurances of QoS during the execution of hosted VNFs.

Compared to our previous work TEMPOS, in TEMPOS4NFV we redesigned the event management functionality to process incoming requests not at the *application* level, but rather directly at the *network* level. TEMPOS4NFV also introduces an innovative QoS-aware multi-site function composition mechanism enabling the creation of VNFs through the coordinated execution of multiple functions available in different Edge sites. Finally, high rates of function activation, reachable by network packet processing, required the integration of optimization mechanisms coupled with levels of QoS to assure adequate performance.

Thanks to its capabilities of offering an effective end-to-end QoS differentiation of Serverless workflows, TEMPOS4NFV mitigates technical and performance constraints that have hindered the adoption of Serverless functions to implement VNFs. TEMPOS4NFV orchestration mechanisms enable seamless coordination and exploitation of different priorities in the workflow invocation stack. The prioritization involves the entire stack of execution from the network layer to the processing of packets and can also span multiple functions deployed on different Edges realizing in this way a *multi-site QoS-aware function composition service*. To ensure effective differentiation of Serverless workloads end-to-end, TEMPOS4NFV integrates and coordinates the heterogeneous hardware and software QoS mechanisms offered at Edge sites. For example, TEMPOS4NFV has the capacity to leverage network slicing mechanisms made available by lower-level protocols, such as Time-Sensitive Networking (TSN). It can also harness real-time processing scheduling policies offered by the operating system, such as the Linux Real-Time Scheduler. To better suit the performance requirements of networking functions, TEMPOS4NFV also exploits *dynamic performance optimizations*, such as Warm Start, in correlating with the QoS workflows.

The architecture of TEMPOS4NFV consists of three *Functional Sections*, each of which offers specific protocols or system mechanisms that can provide and support prioritization. The *Bridging Section* facilitates the exchange of data between the outside world and the platform by using network prioritization and slicing protocols. The *Delivery Section* ensures prioritization in the exchange of events between components of the architecture by relying on both network prioritization and processing mechanisms. Finally, the *Processing Section* manages the execution of functions that process events, leveraging the operating system scheduling algorithms to ensure proper prioritization of computational resources and dynamically activating optimizations to grant required QoS. All these Functional Sections are designed to be independently and transparently deployable on already existing ETSI NFV Management and Orchestration (MANO) [13] orchestrated Edge deployments. TEMPOS4NFV then coordinates with the VIM and VNF manager to dynamically obtain needed resources to run the Serverless Network Function inside provided VNFs.

A. Bridging

In the Bridging Section, we designed the *packet trigger* as the initial architecture component. It intercepts raw network packets, allowing TEMPOS4NFV to operate directly at the network layer. Triggers act as publishers connected to the Message-Oriented Middleware (MOM) and are responsible for receiving and detecting information from the outside world (e.g., requests, external events, and changes in a file system), translating them into internal FaaS events that can be processed by function workflows. In FaaS platforms at the Application Layer events are complex data structures encapsulating information concerning the interaction such as the interacting protocol and contextual information. To avoid overhead caused by the packing of each network packet into a FaaS event, our *TEMPOS4NFV packet trigger* adds a minimal header to each packet differentiating in topics network flows incoming.

TEMPOS4NFV packet triggers are network elements that can be directly addressed as destination routes by other devices and network elements, such as firewalls, routers, and traffic shapers enabling a transparent coexistence with already consolidated physical and NFV solutions. Once captured, the packet is encapsulated in a TEMPOS4NFV event, marked for a specific topic, and forwarded to the MOM through a channel. Triggers are the first QoS-aware differentiating components and they can have any suitable position: they can be located in the same node, either virtual or physical, representing the origin of the events, in a localized perspective, or be chosen as a centralized endpoint for the entire network. A second original element in the Bridging Section is the *sink* that acts as the endpoint of workflows and it is an architecture component that facilitates packet redirection to the final destination. Upon completion of the processing, the output is published to one or more topics associated with sinks. When the event representing the function processing result is received, the sinks extract the final destination from the event header and forward it to the intended recipient.

B. Delivery

The core of the *Delivery Section* is a MOM that supports QoS differentiation of message flows through two abstractions: *TEMPOS4NFV Channels* and *TEMPOS4NFV Topic*. The MOM realizes a QoS-aware Publish/Subscribe communication between triggers and invokers decoupling them in space and time and enabling a more independent and easier scalability of the components. A channel is used to establish a connection between a MOM client, either publisher or subscriber, and the MOM with a specific quality. The MOM allows users to configure the quality available and maps them to the underlying technologies. Once clients are connected to the MOM via one or more specific channels, they can register a TEMPOS4NFV Topic to indicate their desire to be associated with other clients. It is up to the MOM to dynamically create an association among publishers and subscribers of the same topic and to propagate the quality criteria of the channels to the associated topic. TEMPOS4NFV topics represent an abstraction that facilitates the distribution of messages from publishers to subscribers based on specific QoS criteria.

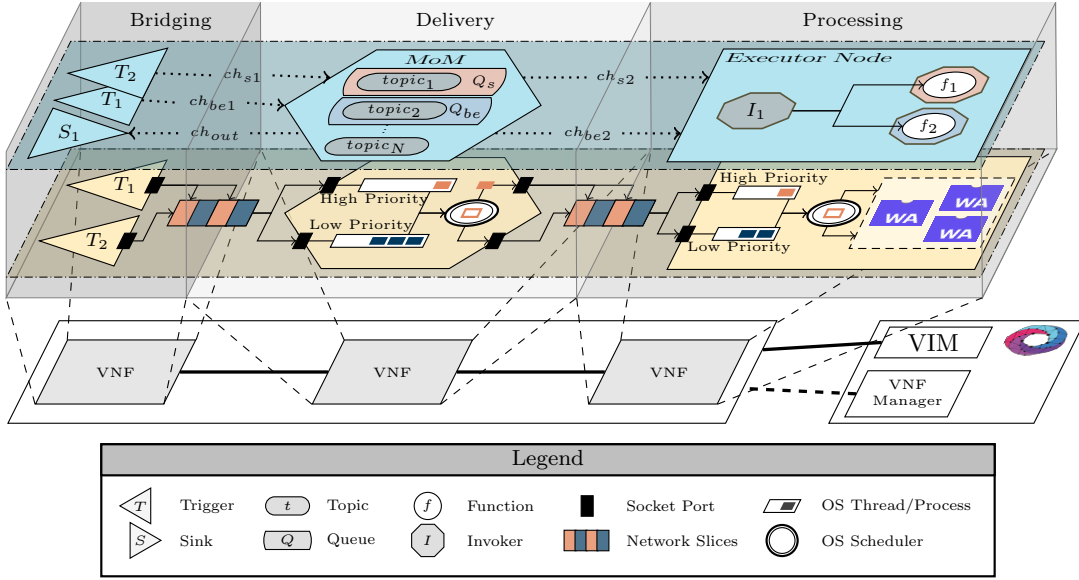


Fig. 1. Multilevel representation of TEMPOS4NFV architecture consisting of three sections (Bridging, Delivery, and Processing) and three conceptual layers (Logical/QoS, System, MEC). The uppermost Logical/QoS layer provides abstractions to the customer and the supported QoS levels. The System layer in the middle encompasses system-specific prioritization mechanisms [12]. Finally, the lowermost MEC layer comprises the deployment and orchestration tools.

By utilizing a MOM as the unique mode of communication between components, communication can be asynchronous and completely location-transparent. That means that each component is unaware of the identity, location, and activation state of the other components that can consume published messages. Such features enable TEMPOS4NFV to exploit heterogeneous resources distributed across any place of the Edge. Asynchronicity and transparency enable components to be either scaled or migrated to different locations without altering the platform. In other words, in our architecture, each site hosts one or more instances of TEMPOS4NFV MOM, and cross-site communication is handled by a *TOPIC-based federation* with other zones. The MOM-based federation of TEMPOS4NFV put the basis to execute QoS-differentiated complex workloads across separated Cloud Edge sites.

C. Processing

In a FaaS platform like TEMPOS4NFV, triggers emit events that are handled by MOM and then processed by executing functions. Within TEMPOS4NFV, *invokers* are the architecture components responsible for managing the execution of functions and they constitute the *Processing Section*. Due to the MOM-centric architecture of TEMPOS4NFV, invokers act as typical subscribers of TEMPOS4NFV topics, thereby benefiting from the transparency and asynchronous processing of events. At the initialization, each invoker establishes multiple channels with the MOM, based on the number of QoS mechanisms supported by deployment nodes. Moreover, invokers can be tailored to different available priority mechanisms, function invocation methods, and function execution environments.

Upon initialization, the invoker becomes responsible for registering one or more customer-specified topics through the workflow definition, serving as the target for executing specific functions. During registration, the invoker and trigger

open a number of channels corresponding to the number of QoS differentiated mechanisms supported by the network technologies. The invoker is also responsible for matching the qualities of the different channels with those supported by the execution environment on which it is running, such as the priority of the operating system scheduler and the scheduling priority of the invoker itself. After receiving any event, the invoker executes the associated function by applying one of the invocation methods supported by the hosting node [12]. At the end of the function, the result of the execution can be published on a topic specified in the workflow definition. TEMPOS4NFV can forward the result to other functions, thus enabling *function-composition*, or to a sink component.

As an optimization, to deal with specific network patterns such as sudden bursts, the invoker monitors the execution performance of each function instance and can decide to operate optimizations based on the QoS of each function requested. As an example, the invoker can decide to retain resources needed to execute more critical functions and/or keep the function active realizing in this way an advanced *QoS-aware Warm Start mechanism*.

The introduction of MOM greatly simplifies the system complexity, enhances flexibility in communications, and grants high-level conceptualization of QoS levels by associating them with topics. Finally, the capacity of transparently orchestrating synchronized and collaborative operations of TEMPOS4NFV architecture components facilitates end-to-end workflow differentiation and resolves resource contention, by promoting broader adoption of Serverless NFV.

As an example (Fig. 1), a typical sequence of steps for activating a Serverless network function involves a trigger that receives a packet and encapsulates it into a well-formed message. The message is then transmitted to the TEMPOS4NFV MOM via the channel ch_{s1} , with the packet marked for publishing in $topic_1$. The MOM identifies the invoker I_1 as a

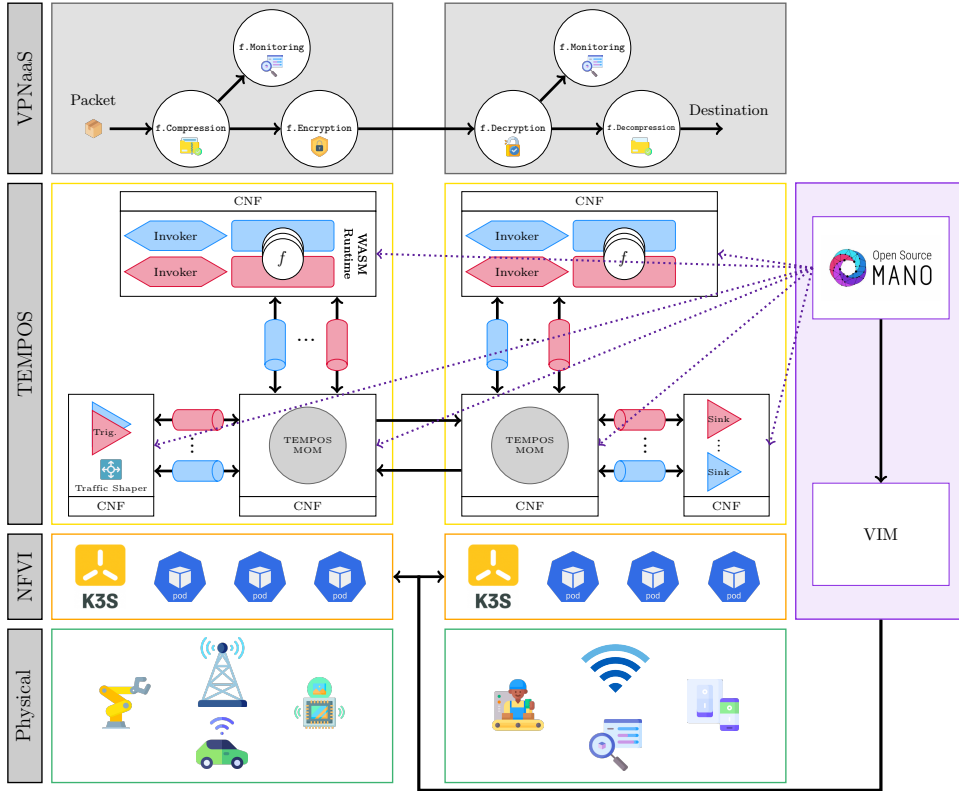


Fig. 2. Our envisioned use case scenario, is characterized by two Edge sites having services and devices that dynamically connect and interact with resources available on both sites. VNFs hosted on a multi-site TEMPOS4NFV installation realize a QoS-aware VPN service while providing observability insights.

subscriber interested in receiving messages published in $topic_1$ and forwards the message through channel ch_{s2} . Then, I_1 consumes the message, activates the function, and forwards the event for processing. After processing, the function returns a result message to I_1 , which publishes it in $topic_{out}$ and forwards it to the MOM via ch_{s2} . Finally, the MOM sends the message to the sink S_1 via ch_{out} , where it is unpacked and sent to its final destination found in the packet header.

IV. CROSS-EDGE SERVERLESS VPN NETWORK FUNCTION

To validate and assess the effectiveness of our approach for robust QoS differentiation of Serverless Workflows, we have deployed and carefully evaluated TEMPOS4NFV in the use case scenario of Fig. 2. In particular, we consider two edge nodes associated with two geographically separated industrial plants interconnected via the public Internet. Deployment and configuration of TEMPOS4NFV components in the two sites are orchestrated through OSM framework with a Kubernetes cluster configured for each site as target Network Function Virtualization Infrastructure. This setup enables the exploitation of the already existing scaling mechanism of OSM and Kubernetes to regulate resources made available to TEMPOS4NFV. The devices, sensors, and users, Mobile Nodes (MN) from now on, can move across the two plants by wireless and 5G connectivity. The connections can be either always-on or established on the fly as needed. MNs are uniquely identified within TEMPOS4NFV which can also update their location. In this deployment, we have hosted in

TEMPOS4NFV a function composition workload implementing a site-to-site VPN Network Function to enable intra-edge communication among devices to demonstrate the capabilities of TEMPOS4NFV to transparently intercept network packets and process them through the use of VNF, while effectively supporting differentiated QoS in the challenging case of different workloads running across multiple sites.

In our VPN service, network packets are intercepted by a Traffic Shaper process and redirected to the TEMPOS4NFV Trigger. Offloading, encryption, and compression functions from devices to edge nodes enable the integration of IoT devices characterized by low computational resources and power constraints. Functions that realize the process of compression and encryption are dynamically activated inside the same VNFs hosting the invokers, only at the transmission of packets, thus reducing the number of processes needed to run at the edge nodes. At the actual state, TEMPOS4NFV does not support any optimized intra-function state sharing. To mitigate that problem, each function deployment has packed the node mapping information the function needs to execute. Efficient and performant state management of FaaS functions is currently an open research issue that we aim to address in future developments [14].

TEMPOS4NFV gives priority to critical workflows by differentiating network and processing resource usage based on desired QoS. TEMPOS4NFV function chaining capabilities also facilitate the parallel execution of monitoring functions during the execution of VPN workflows, by monitoring packet transmission, payload size, and other relevant metrics. Devices

that interact with nodes belonging to other sites can forward packets to a default route endpoint, differentiated by a traffic shaper that forwards them to a specific trigger depending on the QoS associated with the network traffic type. Once compressed and encrypted, network traffic is forwarded to the federated MOM hosted on the second edge node through specific topics. Here, the receivers of packets trigger the execution of a workflow that first decrypts the packets and then decompresses them; the sink then delivers packets to MNs.

A. Experimental Settings

The VPN service presented here is built upon the insights and implementation described in [12] by exploiting the Linux real-time scheduler to prioritize function execution within the WebAssembly (WASM) [12] execution environment, while also leveraging Time-Sensitive Networking (TSN) time-aware shaper (i.e., IEEE 802.1Qbv [15]) to differentiate network quality. In particular, we have configured two services with different levels of priority. The first service, handling the critical flow, is referred to as the *Strict Workflow*; while the other, deemed to lower priority packets, is referred to as the *Best-effort workflow*. This differentiation in services allows more efficient and effective NFV management and ensures packet delivery and processing according to their respective levels of priority.

Reported tests have been conducted over two clusters of 3 nodes each, with 16 physical cores and 64GB RAM interconnected through a 1Gb Relyum RELY-TSN-BRIDGE Ethernet switch. TEMPOS4NFV components have been deployed on each site with the two MOM federated to support multi-site NFV-chaining. In the following two sections, we present collected experimental results. Sec. IV-B first focuses on the behavior in a stationary condition, i.e., constant rate of Packets Per Second (PPS) and results are depicted in Fig. 3; then, we evaluated the behavior under stress by testing with an increasing rate of PPS. Next, Sec. IV-C instead, shows the Warm Start.

B. QoS Differentiation

The TEMPOS4NFV-based VPN service prototype has been extensively tested at a constant rate of 100 PPS with two flows, one with high quality and the other with low, traversing the VPN from one site to the other. The results in Fig. 3 demonstrate that, in normal operation situations, TEMPOS4NFV can strongly differentiate the QoS of workflows, by favoring the execution of the highest-quality one. This differentiation is evident not only in terms of average execution latency but also in terms of jitter.

In addition, to stress the TEMPOS4NFV capability to differentiate traffic, we progressively submit packets to the VPN prototype to achieve resource saturation. The reported results reported in Fig. 4 show that TEMPOS4NFV can maintain end-to-end latency constant for traffic marked as high quality throughout the entire experiment. Differently, the traffic Best QoS progressively degraded due to resource depletion. This fully achieves the QoS differentiation by preserving the Serverless event-driven nature and the zero-scaling features while

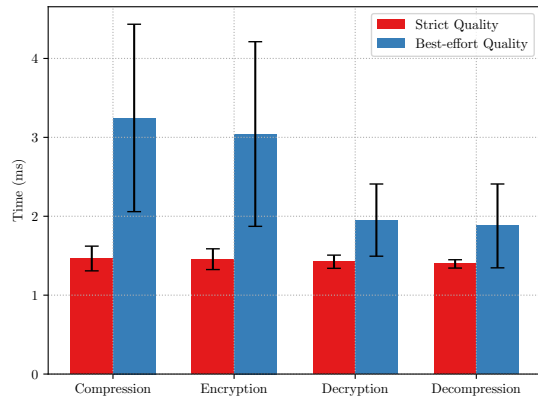


Fig. 3. Performance breakdown of the average execution latency of VNFs realizing the QoS-aware VPN service while a constant rate of packets is processed by both Strict and Best-effort quality flows.

ensuring adequate resources to run the prioritized virtualized functions.

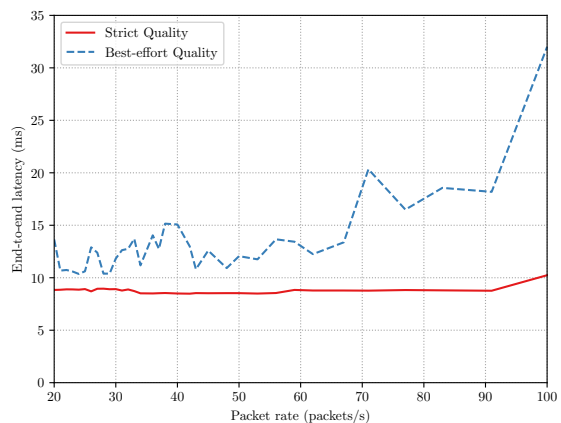


Fig. 4. End-to-end latency experimented when a linearly increasing number of packets starting from 1 packets/s to 100 packets/s are traversing the VPN and differentiated into two Quality: Strict and Best effort.

C. QoS-aware Warm Start

To show the versatility of TEMPOS4NFV, we associate the *Warm Start* feature with the Strict QoS. Specifically, when a high-quality QoS workflow receives successive requests where the time taken to create and execute the function is greater than the time interval between requests plus a given threshold, function instances are retained for multiple successive requests. In the experiment, we challenge TEMPOS4NFV with a progressively increasing packet rate up to 10,000 PPS. Results of Fig. 5 demonstrate that, at low request rates, both functions have adequate resources to execute and coexist. When the request rate increases, TEMPOS4NFV can activate the Warm Start mechanism for strict QoS workloads as specified. Retention of resources and absence of function instance recreation

significantly reduce the execution latency and the overhead associated with the Strict workflow. In contrast, the Best-effort workflow performance rapidly deteriorates, showing an example of a workflow that must compete for resources at each triggering.

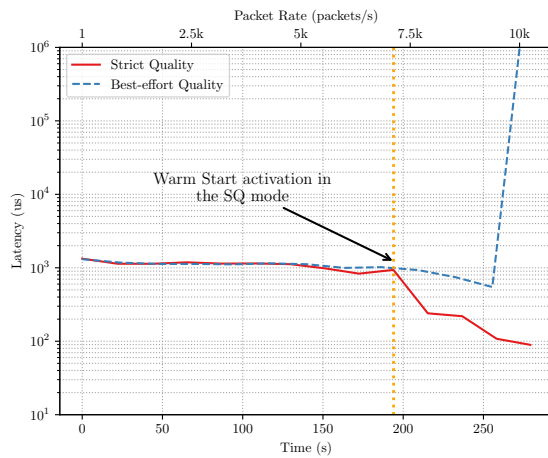


Fig. 5. Execution latency performance of Strict and Best effort workflows when a constantly increasing rate of requests is submitted to both and Warm Start functionality is associated as QoS-aware mechanism to Strict workflow.

V. CONCLUSIVE REMARKS

Serverless Computing is having a fast spreading in many areas, and we adopted it for its provisioning on NFV of a significant impact on unprecedented scaling capabilities and opening up many new perspectives, due to its reduced development and management overhead. However, open technical challenges in orchestration and performance, primarily deriving from the ephemerality and unpredictability of the Serverless function lifecycle, are currently discouraging the wide adoption of this paradigm in many scenarios, including NFV. This paper contributes to the advancement of the state-of-the-art in the field by presenting a middleware capable of supporting and giving priority to critical workloads, by integrating advanced QoS-aware mechanisms and state-of-the-art Serverless technologies. Finally, this work demonstrates that approaches that integrate QoS-aware strategies in the Serverless Computing model could accelerate the application of this paradigm to effectively support NFV provisioning to mobile nodes in Edge Computing scenarios.

REFERENCES

- [1] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on sdn and nfv for next generation iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5706–5712, 2019.
- [2] P. Aditya, I. E. Akkus *et al.*, "Will serverless computing revolutionize nfv?" *Proceedings of the IEEE*, vol. 107, no. 4, pp. 667–678, 2019.
- [3] A. Mampage, S. Karunasekera, and R. Buyya, "A holistic view on resource management in serverless computing environments: Taxonomy and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–36, 2022.
- [4] P. Castro, V. Ishakian *et al.*, "The rise of serverless computing," *Communications of the ACM*, vol. 62, no. 12, pp. 44–54, 2019.
- [5] C. Cicconetti, M. Conti *et al.*, "Toward distributed computing environments with serverless solutions in edge systems," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 40–46, 2020.

- [6] S. R. Chaudhry, A. Palade *et al.*, "Improved qos at the edge using serverless computing to deploy virtual network functions," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10673–10683, 2020.
- [7] M. Savi, A. Banfi *et al.*, "Serverless computing for nfv: Is it worth it? a performance comparison analysis," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 2022, pp. 680–685.
- [8] H. Yu, H. Zhang *et al.*, "Serpens: A high performance faas platform for network functions," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 8, pp. 2448–2463, 2023.
- [9] A. Mohan, H. S. Sane *et al.*, "Agile cold starts for scalable serverless," *HotCloud*, vol. 2019, no. 10.5555, pp. 3 357 034–3 357 060, 2019.
- [10] A. Tariq, A. Pahl *et al.*, "Sequoia: Enabling quality-of-service in serverless computing," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 311–327.
- [11] A. A. Barakabitze, A. Ahmad *et al.*, "5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128619304773>
- [12] A. Garbugli, A. Sabbioni *et al.*, "Tempos: Qos management middleware for edge cloud computing faas in the internet of things," *IEEE Access*, vol. 10, pp. 49 114–49 127, 2022.
- [13] B. Nogales, I. Vidal *et al.*, "Design and deployment of an open management and orchestration platform for multi-site nfv experimentation," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 20–27, 2019.
- [14] A. Sabbioni, A. Bujari *et al.*, "An architectural approach for heterogeneous data access in serverless platforms," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 129–134.
- [15] A. Nasrallah, A. S. Thyagaturu *et al.*, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.

BIOGRAPHY

Andrea Sabbioni received the M.Sc. degree in Computer Science Engineering cum laude from the University of Bologna, Italy. He is currently a Ph.D. student in Computer Science and Engineering at the University of Bologna, Italy. His research interests include Cloud, Fog, and Serverless computing and middleware.

Andrea Garbugli holds an M.Sc. degree in Computer Science Engineering with honors from the University of Bologna, Italy, and is currently a postdoctoral researcher at the same university. His research focuses on ultra-low latency communications and network acceleration in the Edge Cloud.

Luca Foschini (SM, IEEE) received a Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2007 where is an Associate Professor. His interests span from integrated management of distributed systems and services to wireless pervasive computing and scalable context data distribution infrastructures and context-aware services

Antonio Corradi (SM'19) graduated from the University of Bologna, Italy, and received an M.S. degree in electrical engineering from Cornell University, USA. He is currently Full Professor of computer engineering at the University of Bologna. His research interests include all aspects of cloud and middleware, interoperability and innovation, and novel aspects of Cloud Continuum interactions.

Paolo Bellavista (SM'06) received a Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2001, where he is a Full Professor now. His research interests include middleware for QoS management in the cloud continuum, infrastructures for big data processing in industrial environments, and performance optimization in wide-scale and latency-sensitive deployment environments.