SRFog: A flexible architecture for Virtual Reality content delivery through Fog Computing and Segment Routing

José Santos*, Jeroen van der Hooft*, Maria Torres Vega*, Tim Wauters*, Bruno Volckaert* and Filip De Turck*
* Ghent University - imec, IDLab, Department of Information Technology
Technologiepark-Zwijnaarde 126, 9052 Gent, Belgium
Email: josepedro.pereiradossantos@UGent.be

***Abstract*— The advent of softwarized networks has enabled the deployment of service chains of virtual network components on computational resources from the cloud up to the edge. The next generation of use cases (e.g. Virtual Reality (VR) content delivery services) puts even more stringent requirements on the infrastructure, calling for considerable advancements towards fully cloud-native architectures. This paper identifies important challenges for next-generation architectures to support low latency applications throughout their execution life cycle. A flexible architecture named SRFog is presented for the support of VR content delivery in next-generation networks. The approach combines Fog Computing (FC) concepts, an extension of cloud computing, with Segment Routing (SR), which leverages the source routing paradigm. Service Function Chaining (SFC) is also discussed as a major functionality for the proper orchestration of emerging use cases. Early implementations show promising results when deploying container-based VR chains in a flexible FC architecture.**

***Index Terms*—low latency, VR, Fog Computing, Segment Routing, Micro-services, Orchestration**

I. Introduction

The deployment of high-bandwidth and low latency 5G network infrastructures has been driving the digital transformation of network services in Industry 4.0, Smart Cities, Healthcare, or autonomous cars. To overcome the hurdles to arrive at truly End-to-End (E2E) services which meet the even more stringent requirements (e.g. higher bandwidths, lower latencies) of future applications, next-generation (6G) networks have to provide distributed orchestration and management features to integrate a continuum of computing resources [1] with a wide variety of ultra-broadband (radio access and core) and high-precision network links. Bandwidth requirements for Augmented and Virtual Reality (AR/VR), or Holographic Type Communication (HTC) applications will rise well above 1Tbps, while their interactive experiences require sub-millisecond latency [2]. Also, the massive growth of the Internet of Things (IoT) is pushing the boundaries of network architectures by transforming everyday objects into smart connected devices requiring high reliability and low latency guarantees. The deployment of these applications has been hindered by the inefficiency of today's network architectures and protocols to support their requirements [3]. Current architectures lead to low performance and poor Quality of Experience (QoE) to end-users. To ensure low E2E service latency for all emerging use cases, drastic changes in current network architectures need to take place. Several improvements are currently being implemented at the Radio Access Network (RAN) and core alongside novel networking systems incorporating Software-defined networking (SDN) [4] and Network Function Virtualization (NFV) [5] concepts to interconnect resources from the cloud up to the edge, helping to bring low latency services to reality. Nevertheless, novel solutions are needed to bring the latency down to fully support the requirements of emerging use cases.

This paper proposes a flexible architecture to support VR content delivery in next-generation networks called SRFog, leveraging on Fog Computing (FC) [6] and Segment Routing (SR) [7]. FC distributes resources in the network area by bringing processing power, storage, and memory capacity closer to devices and end-users, deploying such resources on edge and fog locations. SR provides higher flexibility and agility by adopting the source routing paradigm. Meaning, a node steers a packet through an ordered list of instructions placed into packet headers. SRFog applies SR to enable fast routing through a service chain deployed at different infrastructure levels (i.e. edge, fog, cloud). The concepts of Service Function Chaining (SFC) and Micro-services are also adopted in the SRFog architecture since both technologies are promising approaches to enable low latency service delivery in future networks.

The next section revisits the state-of-the-art on low latency service delivery in softwarized networks and identifies important challenges. Afterwards, the SRFog architecture is presented in Section III alongside two main topics: the allocation of container-based VR chains and the integration of SFC with SR to enable fast and scalable orchestration of VR applications. Section IV focuses on open challenges and future directions while concluding remarks are presented in Section V.

II. State-of-the-Art & Challenges

Several steps in the service execution causing increased delays have to be recognized as potential barriers to low E2E service latency. First, a fast and scalable E2E connection setup is required, typically across access and core network domains. Distributed (hierarchical) SDN architectures are considered better candidates than current centralized control architectures [8]. However, further research is needed to understand how such architectures can incorporate local or global network measurements and analysis to steer the message routing, and how service-level objectives can be enforced in the network. Along the network path, fog-cloud resources need to be set up to execute several micro-services of a service chain, leading to flexible deployments supporting low latency service delivery [9]. The SRFog architecture is detailed in Section III-A. Current service allocations are mostly orchestrated in a static, centralized manner and typically focus on optimizing energy and cost efficiency [10]. Further work is needed to support dynamic resource provisioning, optimize transport and computing latency, and take specific requirements of future low latency applications into account. Section III-B features this topic.

In turn, once flows are set up, network paths are chosen and service components installed, low latency communication protocols are necessary to efficiently deliver content to end-users. New IP protocols [11] have also included latency-aware features in these architectures, such as Deterministic Networking (DetNet), Time-Sensitive Networking (TSN) and SR over IPv6 (SRv6) [12]. DetNet assures bounds on latency, delay variation, and packet loss for prioritized services. It can support high reliability and low latency services such as autonomous driving and factory automation, by allowing 'soft' slicing that considers the desired service performance. TSN provides time synchronization and time-aware, reliable scheduling for real-time communication with non-negotiable time boundaries for E2E transmission latencies, while SRv6 replaces tunneling with IP options and enables steering of packets belonging to a flow via a set of instructions. These are referred to as (topological or service-based) instructions or 'segments'. These segments can (i) enforce flows through specified strict or loose paths, (ii) allow the selection of a link, buffer, or a Quality of Service (QoS) treatment within a node and (iii) direct a packet through an ordered sequence of services, enabling flexible and fast service chaining. Section III-C discussed the integration of SR and SFC.

Besides underlying IP layer protocols, application and transport layer protocols need to be tuned towards low latency services. Many Web-based applications employ HTTP (over TCP), which supports features to decrease latency, such as request multiplexing and pipelining, and server push (since HTTP/2). Recent developments towards HTTP/3 fix head-of-line blocking by using the transport protocol QUIC, thus potentially reducing start-up latency. While the abovementioned techniques aim to reduce delays, application- or network-specific optimizations can also mask or compensate for delays. By gathering, analyzing, and learning from network and application data, context changes can be predicted and actions executed beforehand accordingly. Changing network characteristics, user movements, data access patterns, or sensor locations can therefore be dealt with appropriately by setting up new routes or selecting and transporting different content at different quality beforehand. In addition, the integration of intelligence at the edge will lead to Machine Learning (ML)-driven networks able to support highly dynamic management updates under varying network circumstances, meeting the requirements of cloud-native applications and services over the continuum of virtual resources.

III. SRFog: Towards latency-aware SFC allocation

This section presents the SRFog architecture for emerging use cases in next-generation networks based on FC and SR. SFC allocation for fog-cloud infrastructures is also discussed in detail. Furthermore, the integration of SFC and SR is described, in which SRv6 can help achieve fast routing in service chains through the source routing paradigm. Finally, a container-based VR SFC has been designed to serve as an example of how service allocations are performed in SRFog.

A. SRFog Architecture

The advent of novel architectural paradigms enables the deployment of service chains on computational resources from the cloud up to the edge, providing several benefits such as low latency and mobility support. Fig. 1 presents a high-level view of the SRFog network infrastructure, showing its key architectural concepts based on FC and the micro-service paradigm. SRFog provides resources throughout the network area, including the core and the edge. In contrast to centralized clouds, fog and edge nodes are distributed across the network to act as intermediate nodes between end devices and the cloud. These fog nodes or edge locations bring processing power, storage, and memory capacity closer to devices and end-users. The main difference between edge and fog nodes is where the processing takes place since the first usually is the end device itself while fog nodes are intermediate locations close to end devices. All network locations are Points of Presence (PoPs) consisting of networking equipment such as routers, switches, and computing resources (e.g. servers).

In addition, SRFog follows the micro-service pattern [13]. An application is decomposed into a set of loosely coupled services that can be developed, deployed, and maintained independently. Each service is responsible for a single task and communicates with the other services through lightweight protocols. Containers are currently the most promising alternative to the traditional monolithic application paradigm, where almost everything is centralized and code-heavy. Due to their low resource usage and high portability, containers are also the main alternative to conventional Virtual Machines (VMs). With their massive adoption, orchestration solutions for containerized services have been developed by IT companies and open-source communities. Among those, the most widely used today is Kubernetes [14], an open-source orchestration platform for automatic life cycle management of containerized applications. All SRFog nodes are based on the Kubernetes architectural model shown in Fig. 2. Kubernetes follows the master-slave model, where at least one master node manages containers across multiple worker nodes (slaves). In SRFog, master nodes are deployed in cloud locations, while worker

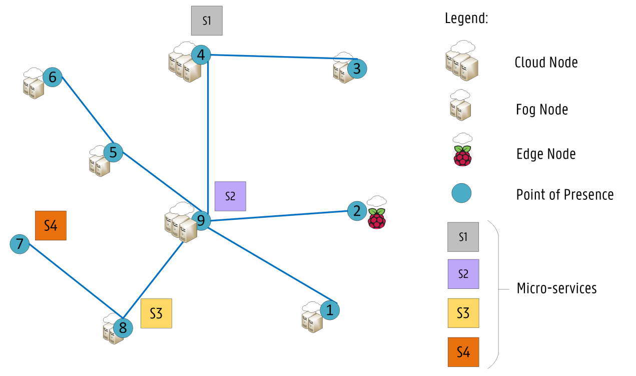


Fig. 1: High-level view of the SRFog network infrastructure.

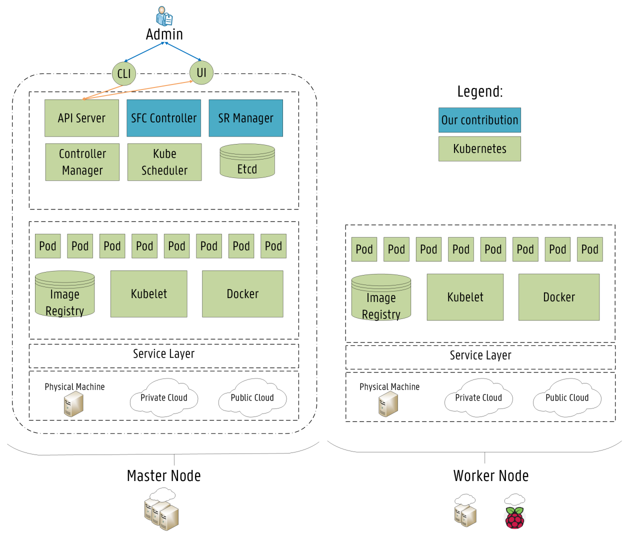


Fig. 2: The SRFog Kubernetes-based architectural model.

nodes are set up in fog or edge locations. Cloud nodes typically have more computing power to operate all software elements composing a master node. Kubernetes already provides several components (e.g. API server, Kubelet, Controller manager) to handle the complete life cycle workflow of containerized applications. SRFog extends the Kubernetes architecture with two software components, an SFC controller and an SR Manager. The next section describes how the SFC controller performs container-based service chain allocations.

B. Container-based SFC allocation

Although Kubernetes employs containers as the underlying mechanism to deploy micro-services, additional layers of abstraction exist over the container runtime environment to provide scalable life cycle orchestration features. In Kubernetes, micro-services are often tightly coupled into a group of containers. This group is the smallest working unit in Kubernetes, called a *Pod*, that represents the collection of containers and volumes (storage) running in the same execution environment [14]. Based on service requirements and available resources, the master schedules a pod on a specific node. Then, the assigned node pulls the container images from the image registry if needed and coordinates the necessary operations to launch the pod. The Kube-scheduler (KS) assigns a node to each pod. The KS is the default scheduling component in the Kubernetes platform, responsible for deciding where a specific pod should be deployed. Although Kubernetes supports policy-rich and topology-aware features, the KS merely considers the number of requested resources on each host, thus only optimizing the scheduling for hardware cost (i.e. CPU and RAM usage rates) without considering latency. Therefore, in our previous work, a latency-aware scheduler for Kubernetes has been proposed in [15] as a Proof of Concept (PoC). An extension followed in [16], presenting an SFC controller for the efficient provisioning of container-based service chains focused on optimizing resources and reducing E2E latency. The SFC controller extends Kubernetes functionalities by enabling the allocation of container-based service chains based on latency and location. Results have proven that the proposed approach can significantly reduce network latency while respecting bandwidth guarantees. The SFC controller is included in the SRFog architecture to deploy service chains based on E2E latency reduction. Recently, a Mixed-Integer Linear Programming (MILP) model [10] addressed fog-based SFC allocation considering service chaining concepts (e.g. services with different replication factors) and multiple optimization objectives. Results have shown differences between the assessed provisioning strategies (i.e. low latency vs high energy efficiency). Thus, as early work, the previously presented MILP formulation will be extended and adapted for VR content delivery in the SRFog architecture. The model will serve as a benchmark for future heuristics implemented in the SFC controller. The integration of SFC and SR is explained next.

C. Service Function Chaining based on Segment Routing

Kubernetes provides a feature called *Service*, which is an abstract way to define a logical set of pods and expose applications running on them as a network service [14]. By applying this abstraction, there is no need for a service discovery mechanism since pods have their own IP address and a single Domain Name System (DNS) name is assigned to a set of pods, which makes load balancing a straightforward process across them. The rationale behind this abstraction comes from the pods' volatility as they may be terminated, meaning that pods running at a certain moment may be different than those providing the service a few days later without requiring user awareness. However, this means that Kubernetes networking is currently reliant on assigning an IP address to each pod, leading to several issues (e.g. containers IP dependencies, IPv4 address exhaustion). Additionally, Kubernetes does not provide any networking solution but instead relies on third-party plugins to handle container networking. Thus, a novel component named SR Manager is presented to enable SRv6 for container networking in SRFog. SR leverages the source routing paradigm, in which a node steers a packet through an ordered list of instructions, called segments. A segment can include several types of instructions (e.g. topological or

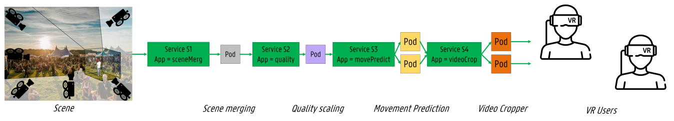


Fig. 3: The envisioned container-based VR service chain.

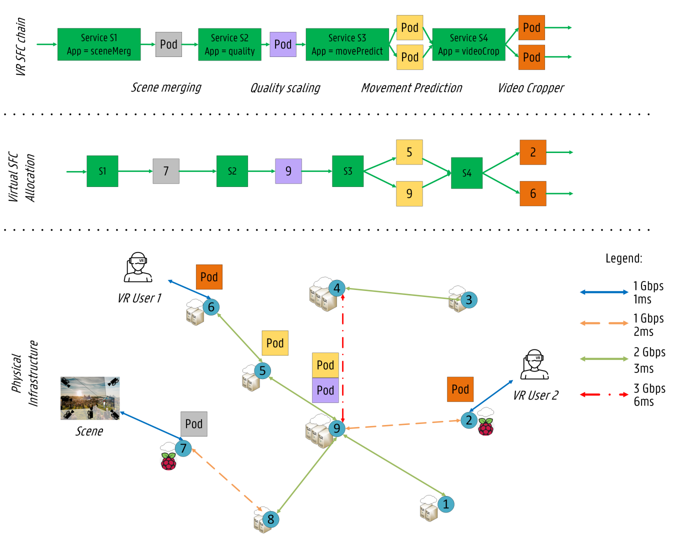
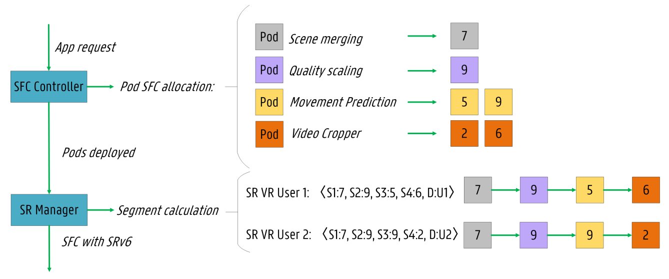


Fig. 4: High-level view of an illustrative SFC allocation.



SR VR User 1: ⟨S1:7, S2:9, S3:5, S4:6, D:U1⟩

SR VR User 2: ⟨S1:7, S2:9, S3:9, S4:2, D:U2⟩

Fig. 5: Example to illustrate the application instantiation process in SRFog.

service-based information), which are then placed as path state information into a packet header at an ingress node. Several segments can create unconstrained network paths represented by the so-called Segment Lists. Information flows from packet headers, making nodes stateless and significantly reducing the complexity of forwarding tables, simplifying traffic engineering and management across network domains. SR is highly responsive to network changes, making networks more agile and flexible. The main goal of integrating SR and SFC is to encode the service chain path in packet headers. Recent work addresses the integration of SR concepts into service chaining [17]. Next, a container-based VR chain has been designed to highlight the integration of SFC and SR in SRFog.

D. Virtual Reality (VR) content delivery use case

Fig. 3 shows an example of a container-based VR SFC while Table I shows the correspondent deployment requirements. Latency is the most important requirement of VR applications since human perception requires accurate and smooth movements. High latency leads to poor VR experiences and contributes to motion sickness [18]. VR developers and industries agree that application round-trip latency should be less than 20 ms for the Motion-To-Photon (MTP) latency to become imperceptible [19]. Current locally deployed VR systems are fine-tuned to meet the 20 ms threshold in their Head-Mounted Displays (HMDs). Based on the presented VR SFC, Table II presents estimated latency values for all components. Sensing and motion capture take around 1 ms, while the network infrastructure is the main contributor to latency in current deployments. Data needs to traverse the network being queued, processed, and transmitted by an increasing number of routers. This procedure adds notable delays depending on the state of the network, making it hard to predict. For instance, reaching centralized cloud-based VR services would take more than 40 ms, resulting in high E2E latency which would disrupt immersion and create discomfort for end-users. Computing and display further contribute to this delay. Even considering high-end hardware and novel processing techniques, sampling, rendering, and displaying will take 5 to 8 ms [19]. When adopting SRFog, E2E latency between 11 - 20 ms is expected, overcoming the current limitations of centralized clouds by providing services closer to end-users as movement prediction services can reduce the network latency by proactively fetching content. VR components hosted in a fog-cloud infrastructure, combined with forthcoming display and computing technologies, can achieve comparable latency to current local deployments. Another relevant requirement is the network bandwidth, the throughput required to stream video feeds to a user. The resolution of a VR immersive video viewed using an HMD needs to reach the detail the human retina can perceive. 8K quality and above is necessary for photo-realistic VR, as a 4K VR video only has 10 pixels per degree, equivalent to 240p on a TV screen [18]. Also, smooth experiences require a high refresh rate of at least 60 fps. Researchers consider that a network throughput of at least 400 Mbps is needed to deliver immersive VR video experiences. Thus, scheduling mechanisms should not consider E2E latency as the only factor in the service allocation. Network bandwidth and resource usage also play a key role in optimizing VR content delivery.

Fig. 4 depicts a high-level view of the VR SFC allocation, while Fig. 5 presents the application instantiation process based on the SFC controller and the SR Manager components. As previously stated, the SFC controller optimizes pod allocations based on E2E latency, while the SR Manager steers the traffic via SRv6 policies. These policies follow the segment list instructions calculated based on pod allocations. For each user accessing the service chain, a segment list is established with the optimized sequence of service-node combinations that their packets should follow to access the deployed chain. These combinations follow developer guidelines instructed via service slots. This information is available via pod configuration files [16]. For instance, since each user requires one slot, for the *sceneMerg* pod with ten service slots, each pod

TABLE I: Deployment properties of the VR Use Case.

Application	Component	Pod Name	Chain Position	Target Location	Min. Bandwidth (Mbps)	CPU (cpu)	RAM (Mi)	Service Slots	Replication Factor
VR	Sensing	sceneMerg	1	Edge/Fog	750.0	1.0	1.0	10	1
	Computing	quality	2	Any	500.0			8	1
	Computing	movePredict	3	Any	500.0			6	2
	Display	videoCrop	4	Edge/Fog	400.0			10	2

TABLE II: The expected latency of VR components in different approaches (in ms) [18], [19]

Component	Local VR	Cloud-based VR (centralized)	VR chain in SRFog (distributed)
Sensing	1	1	1
Network	2 (local)	40 (cloud)	2 - 7 (edge/fog)
Computing	5 - 7	5 - 7	5 - 7
Display	5	5	3 - 5
Total	13 - 16	51 - 53	11 - 20

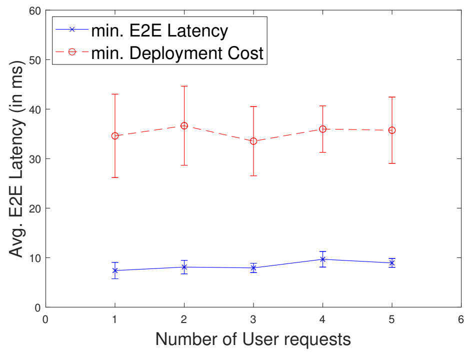


Fig. 6: The E2E latency for the VR use case for both strategies.

instance can host ten users. The SR Manager defines the path for each user based on these guidelines. Then, monitoring services collect data on the deployed pods (e.g. resource usage, bandwidth usage) to help the SRFog adapt the predefined paths if needed due to increased demand. To evaluate the impact of E2E latency, two opposing strategies have been assessed for the VR chain allocation in SRFog: reducing E2E latency and minimizing deployment cost. Both objectives are based on the previously mentioned MILP formulation [20]. A set of applications A composed of micro-services S are allocated on nodes N. Each application a has a given SFC identifier ID. All micro-services have a maximum number of replicas given by β. The replication factor for a particular micro-service s for the application a with the SFC identifier id is given by $\beta_{a,id,s}$. Each micro-service s has a CPU and a memory requirement represented by ω_s (in cpu) and γ_s (in Mi) respectively. Also, each micro-service s has a minimum bandwidth requirement represented by δ_s (in Mbps). A binary placement matrix P is used to represent in which node n, the replica β_i of a micro-service s has been allocated. Additionally, ϖ_n corresponds to the associated node weight (e.g. fog nodes have a lower weight than cloud nodes). The deployment cost minimization can be expressed as shown in Eq. 1. The MILP has been extended to consider E2E latency from the scene to each user. Due to space restrictions, its formulation has not been included, but it will be addressed as future work. Both objectives have been executed 20 times considering confidence intervals of 95%. As shown in Fig. 6, differences of 25 ms between the two strategies are already expected for one user request.

$$\sum_{a\,\varepsilon\,A}\sum_{id\,\varepsilon\,ID}\sum_{s\,\varepsilon\,S}\sum_{\beta_i\,\varepsilon\,\beta}\sum_{n\,\varepsilon\,N} P^{a,id}_{s,\beta_i}(n)\times\varpi_n\times\omega_s\times\gamma_s\times\delta_s \quad (1)$$

E. Summary

In recent years, FC architectures demonstrated the potential to enable emerging use cases through their significant benefits compared to centralized infrastructures (e.g. mobility support, interactivity, backhaul bandwidth conservation). Container-based VR service components (e.g. quality scaling, movement prediction) allocated in the SRFog architecture at a proper location will help achieve low latency service delivery to provide immersive VR experiences to end-users comparable to current VR deployments. This work presents an integrated solution for VR content delivery in next-generation networks that goes beyond existing research that only addresses a subset of the requirements. The SRFog architecture aims to solve resource allocation and routing by optimizing the allocation of VR service chains based on E2E latency reduction and applying SRv6 to provide a flexible and scalable approach to steer traffic in container-based service chains, respectively.

IV. Open Challenges

Open challenges remain before relying on current management and orchestration practices to solve low latency service delivery in next-generation networks. Networks and services keep evolving, with new protocols and technologies introduced to tackle individual problems and improve overall QoS and cost-efficiency. Reinforcement Learning (RL) methods have already proven their potential applicability to resource allocation issues for years. However, their performance is deeply interconnected with the RL system setup. The interactions between the RL agent and the environment affect the performance of these algorithms. The problem complexity and constraints can grow exponentially depending on the considered use case (e.g. number of PoPs, number of services), leading to a computationally hard problem. The advantage of RL over centralized modeling approaches is that no information is needed about the problem beforehand. As a

PoC, an RL environment has been developed to teach agents how to allocate service chains in FC, performing comparably to state-of-the-art ILP formulations while being more scalable [20]. The existing RL PoC will be adapted for the VR use case. Another challenge is mobility support. No generic approach has yet been proposed to deal with service guarantees when end-users or devices move in the network. FC aims to solve these challenges focused on IoT use cases. SRFog adopts FC concepts to position itself as an alternative, addressing the mobility of multiple devices (e.g. VR users and sensors). The evolution beyond 5G will introduce new use cases requiring ultra-high bandwidth. Future VR use cases require real-time operations in a fully-immersive environment, in which throughput availability will play a major role due to its data-hungry nature. Also, the network would benefit from novel mechanisms to decide the best routes for the packets given context information (e.g. user location, mobility patterns) and requested user QoE. SR is a solution towards fast routing in highly complex service chain structures, while providing high resilience guarantees against attacks, unexpected failures, and incorrect sources of information.

V. Conclusions & Future Work

Many next-generation applications, deployed as micro-service chains on softwarized infrastructures, have latency requirements of a few milliseconds, which need support throughout their execution life cycle. Fast flow setup, low latency transport, and intelligent application-level predictions are indispensable. Management and orchestration functionalities need to react timely to changing network characteristics, computational load, and usage patterns. This paper presents a flexible architecture named SRFog for emerging use cases based on FC and SR. SRFog aims to optimize the allocation of service chains based on E2E latency reduction and applying SRv6 to provide a flexible and scalable approach to steer traffic in container-based service chains. An evolution from static, modeling-based approaches towards highly dynamic and scalable techniques, fed by pervasive network and application telemetry, shows promising results. Additional research is required to optimize orchestration systems and successfully incorporate quality of experience parameters. As future work, the development of an RL environment is planned to evaluate the performance of applying RL to allocate container-based VR services in SRFog and potentially incorporate RL agents in the SFC controller.

Acknowledgment

This research is the result of a collaborative project between Huawei and Ghent University, and funded by Huawei Technologies, China. Jeroen van der Hooft and Maria Torres Vega are funded by the Research Foundation Flanders (FWO), grant numbers 1281021N and 12W4819N.

References

[1] (2020) White paper on 6g networking. [Online]. Available: https://www.6gchannel.com/portfolio-posts/6g-white-paper-networking/

[2] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6g networks: Use cases and technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, 2020.

[3] M. F. Zhani and H. ElBakoury, "Flexngia: A flexible internet architecture for the next-generation tactile internet," *Journal of Network and Systems Management*, pp. 1–45, 2020.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[5] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[6] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Fog computing: Enabling the management and orchestration of smart city applications in 5g networks," *Entropy*, vol. 20, no. 1, p. 4, 2018.

[7] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 464–486, 2018.

[8] H. K. Ravuri, M. T. Vega, T. Wauters, B. Da, A. Clemm, and F. De Turck, "An experimental evaluation of flow setup latency in distributed software defined networks," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 432–437.

[9] S. R. Chowdhury, M. A. Salahuddin, N. Limam, and R. Boutaba, "Re-architecting nfv ecosystem with microservices: State of the art and research challenges," *IEEE Network*, vol. 33, no. 3, pp. 168–176, 2019.

[10] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards end-to-end resource provisioning in fog computing over low power wide area networks," *Journal of Network and Computer Applications*, vol. 175, p. 102915, 2021.

[11] R. Li, K. Makhijani, and L. Dong, "New ip: A data packet framework to evolve the internet," in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2020, pp. 1–8.

[12] K. Samdanis and T. Taleb, "The road beyond 5g: A vision and insight of the key technologies," *IEEE Network*, vol. 34, no. 2, pp. 135–141, 2020.

[13] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice architecture: aligning principles, practices, and culture*. " O'Reilly Media, Inc.", 2016.

[14] K. Hightower, B. Burns, and J. Beda, *Kubernetes: up and running: dive into the future of infrastructure*. " O'Reilly Media, Inc.", 2017.

[15] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards network-aware resource provisioning in kubernetes for fog computing applications," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 351–359.

[16] ——, "Towards delay-aware container-based service function chaining in fog computing," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[17] A. AbdelSalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based nfv infrastructure," in *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2017, pp. 1–5.

[18] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "Vr is on the edge: How to deliver 360 videos in mobile networks," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, 2017, pp. 30–35.

[19] M. T. Vega, C. Liaskos, S. Abadal, E. Papapetrou, A. Jain, B. Mouhouche, G. Kalem, S. Ergüt, M. Mach, T. Sabol *et al.*, "Immersive interconnected virtual and augmented reality: A 5g and iot perspective," *Journal of Network and Systems Management*, pp. 1–31, 2020.

[20] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Resource provisioning in fog computing through deep reinforcement learning," in *2021 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2021, pp. 1–6.