Efficient Orchestration of Service Chains in Fog Computing for Immersive Media

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*— Immersive media services, such as Augmented and Virtual Reality (AR/VR) are getting significant attention in recent years with the promise of bringing immersive experiences to end users. However, despite the remarkable advances in the field, AR/VR applications are mostly local and individual experiences. The main obstacle between current technology and future remote, multi-user AR/VR applications is the stringent end-to-end (E2E) latency requirement, which cannot exceed 20 ms to avoid motion sickness. Emerging AR/VR services put even more pressure on current network infrastructures, calling for considerable advancements toward fully cloud-native architectures. Cloud-based VR services, where participants can virtually interact across vast distances, remain a distant dream. Several challenges still arise concerning the deployment and management of VR services. This paper presents a Mixed-Integer Linear Programming (MILP) formulation for the efficient orchestration of VR services in fog-cloud infrastructures. The model considers Fog Computing (FC), an extension of cloud computing, and Segment Routing (SR), which leverages the source routing paradigm. The evaluation of realistic VR container-based service chains shows that deploying VR components hosted in a fog-cloud infrastructure can satisfy the 20 ms latency boundary.**

***Index Terms*—Resource Allocation, Service Function Chaining, Virtual Reality, Fog Computing, Segment Routing, ILP**

I. Introduction

Recently, a huge interest has grown over the commercialization of immersive media services, such as Augmented and Virtual Reality (AR/VR) applications [1], Free-Viewpoint Video (FVV) [2] and 360-degree video services [3]. Vertical markets such as healthcare and manufacturing are adopting immersive services to increase efficiency and productivity through virtual training systems and remote collaboration between workers [4]. In the consumer domain, immersive entertainment experiences (e.g. gaming, music concerts, sporting events) are also gaining significant attention.

Despite recent remarkable technological advancements, VR experiences remain mostly individual and local due to the expensive hardware (e.g. high-end PC) and stringent network requirements (e.g. millisecond latency). Cloud-based VR services, where users can virtually interact with each other across long distances, remain a distant dream [1]. The great barrier standing between current technology and such remote multi-user immersive applications is the stringent end-to-end (E2E) latency requirement. VR developers and industries agree that application round-trip latency needs to remain below 20 ms for the Motion-To-Photon (MTP) latency to become imperceptible [5]. High latency leads to poor VR experiences and motion sickness [6]. Current locally deployed VR systems are fine-tuned to meet the 20 ms threshold in their Head-Mounted Displays (HMDs). However, achieving this 20 ms for remote experiences is quite challenging.

Emerging VR services placed even more pressure on current cloud-based infrastructures, calling for solutions with in-network computation. Applications are then split into microservices and deployed close as possible to end users since centralized and monolithic cloud deployments cannot meet latency requirements. Adopting fog-cloud infrastructures can overcome the shortcomings of centralized clouds, reducing the network latency to deliver low latency service delivery in a continuum of virtual resources [7]. In [8], SRFog has been presented, an architecture adopting Fog Computing (FC) and Segment Routing (SR) for VR content delivery. In [9], an IoT service placement approach for FC environments has been proposed. This paper builds further on both works by presenting a Mixed-Integer Linear Programming (MILP) model for VR service orchestration in SRFog. Several additions have been made to the previous modeling [9] in terms of microservice replication, latency sources, among others. The goal is to optimize traffic flow in container-based service chains since SR provides a high level of flexibility when designing specific chain paths for different User Groups (UGs). These paths are placed into packet headers as an ordered list of instructions, simplifying traffic engineering and delivering lower E2E latency. VR service chains based on live scenarios have been designed to evaluate the model. Results show that deploying VR components hosted in a fog-cloud infrastructure, combined with view prediction services, satisfies the MTP latency threshold of 20 ms.

The remainder of the paper is organized as follows. Related work is discussed in Section II. Section III presents the MILP model for VR content delivery. Section IV introduces the VR service chains. Then, the evaluation setup is described in Section V, which is followed by the results in Section VI. Finally, conclusions are presented in Section VII.

II. Related Work

Resource allocation for VR services has recently gained significant attention in the field of FC. In [10], QoE models have been proposed for Quality of Service (QoS) assessment

978-3-903176-36-2 ©2021 IFIP

in FC regarding system and users. The QoE has been formulated as a joint resource allocation problem under different transmission rates. A dynamic algorithm based on the shortest path tree has also been presented. Evaluations have shown high QoE performance but did not consider latency implications. In [11], a framework for 360-degree video distribution in an FC environment has been presented. Their approach follows microservice design patterns across three computing tiers: cloud, edge, and fog. Results have shown benefits regarding deployment costs and bandwidth consumption. In [12], a Fog Radio Access Network (F-RAN) framework for mobile VR delivery has been proposed. The authors aim to increase spectral efficiency by maximizing the average tolerable delay while meeting high transmission rate requirements. Results have shown that local caching and computing improve the average delay. In [13], FC has been presented as an immersive media enabler. The authors propose several service scenarios and identify key technologies (e.g. F-RAN) for the support of VR content delivery. Service Function Chaining (SFC) nor SR have been studied. Recently, in [14], an allocation mechanism for VR content with QoE support has been presented. The work focuses on deploying microservices based on transmission delays, migration time, and resource usage. Results have demonstrated that reducing the number of stalls, stall duration and buffering time improves the delivered QoE to end users. However, the complete E2E path is not addressed.

With the advent of the Internet of Things (IoT), resource allocation in FC infrastructures has also become an important research topic. In [15], two placement strategies in FC based on matching game algorithms have been introduced. The first strategy considers SFC concepts and the corresponding ordered sequence of services requested by each application. The second one overlooks the SFC structure, to lower the computation complexity without losing performance. Results have shown the increased performance of the stated methods. In [16], bayesian learning techniques have been proposed for resource allocation in FC. The authors focus on the dynamic scaling of resources according to the current network demand. Simulations have shown reduced costs and minimum delay violations. In [17], a resource allocation approach based on dynamic deadline-based requirements has been presented. It hierarchically provisions resources by considering dynamic changes in user requirements and the limited available resources in fog nodes. Simulations have shown improvements concerning data processing time, allocation costs, and network delay compared to existing methods.

Most works have not addressed realistic E2E latency demands. Also, only a few studies have analyzed service chaining or container-based applications. Latency-aware approaches did not cover all latency sources as we consider in our model. These works focus on reducing the network latency in the SFC. In contrast, our model considers the complete E2E path: VR scene latency (content location), network latency, microservice execution time, and user-perceived latency (user location). To the best of our knowledge, our work goes beyond the current literature by considering all these latency sources.

TABLE I: Input variables of the MILP model.

Symbol	Description
N	The set of nodes on which microservice instances are executed.
A	The set of all VR applications. Each application consists of a set of different microservices.
S	The set of all microservices.
ID	The set of SFC Identifiers (IDs).
U	The set of User Groups (UGs).
$\underline{VR}$	The set of VR scenes (i.e. content).
L	The set of locations where applications are deployed.
$\Phi_{u,a}$	The user assignment matrix. If $\Phi_{u,a} = 1$, the UG u accesses the application a.
ρ_a	The maximum number of UGs for an application.
ρ_s	The maximum number of UGs for a microservice.
$\underline{\kappa_s}$	The computing cost of each microservice s.
$\underline{\lambda_u}$	The UG u association cost for the assigned application.
β	The maximum replication factor for each microservice.
υ_s	The SFC first position indicator. If $\upsilon_s = 1$, the microservice s is the first microservice in the service chain.
ι_s	The SFC last position indicator. If $\iota_s = 1$, the microservice s is the last microservice in the service chain.
$I_{a,s}$	The Instance matrix. If $I_{a,s} = 1$, the microservice s belongs to application a.
$\underline{\varpi_n}$	The cost (in units) of the node n.
$\underline{\Omega_n}$	The total CPU capacity (in cpu) of the node n.
Γ_n	The total memory capacity (in Mi) of the node n.
Δ_n	The bandwidth capacity (in Mbps) of the node n.
ω_s	The CPU requirement (in cpu) of the microservice s.
γ_s	The memory requirement (in Mi) of the microservice s.
δ_s	The bandwidth requirement (in Mbps) of the microservice s.
B_{n_1,n_2}	The bandwidth matrix indicates the available capacity (in Mbps) between the node n_1 and the node n_2.
τ_{n_1,n_2}	The node matrix indicates the latency (in ms) between the node n_1 and the node n_2.
τ_{l_1,l_2}	The location matrix indicates the latency (in ms) between the location l_1 and the location l_2.
$\tau_{n,l}$	The node location matrix indicates the latency (in ms) between the node n and the location l.
$\tau_{u,l}$	The user location matrix indicates the latency (in ms) between the UG u and the location l.
$\tau_{u,n}$	The user node matrix indicates the latency (in ms) between the UG u and the node n.
$\underline{\tau_{vr,n}}$	The scene node matrix indicates the latency (in ms) between the scene vr and the node n.
C_{s_i,s_j}	The communication matrix indicates the minimum bandwidth (in Mbps) required between the microservices s_i and s_j.
$E_{n,l}$	If $E_{n,l} = 1$, the node n is at location l.
$E_{u,l}$	If $E_{u,l} = 1$, the UG u is at location l.
$\underline{E_{vr,l}}$	If $E_{vr,l} = 1$, the scene vr is at location l.
α_{s_i,s_j}	The service matrix. If $\alpha_{s_i,s_j} = 1$, the flow bandwidth between the microservices s_i and s_j is guaranteed.

III. Toward latency-aware SFC allocation for VR in fog-cloud infrastructures

A. Model description & Variables

The MILP model significantly extends the authors recent work [9]. Several additions have been made to the model regarding service chaining and E2E latency. The most relevant ones are the following:

- Inclusion of novel decision variables: user chain associations, execution time of microservices and SFC latency.
- Two different objectives: *MIN* deployment costs and *MIN* E2E latency.

Table I shows input variables, while Table II presents decision variables. All variables added to previous work have been underlined: five input variables and eight decision variables.

TABLE II: Decision variables of the MILP model.

Symbol	Description
$G_{a,id}$	The acceptance matrix. If $G_{a,id} = 1$, the application a with the SFC ID id is allocated.
$G_{a,id,s}$	The microservice acceptance matrix. If $G_{a,id,s} = 1$, the microservice s for the application a with the SFC ID id is allocated.
$G_{u,a,id}$	The user accept. matrix. If $G_{u,a,id} = 1$, the UG u joins application a with SFC ID id.
$\beta_{a,id,s}$	The replication factor for the microservice s belonging to application a with SFC ID id.
$P^{a,id}_{s,\beta_i}(n)$	The placement matrix. If $P^{a,id}_{s,\beta_i}(n) = 1$, the replica β_i of microservice s is executed on node n for the application a with SFC ID id.
U_n	The node utilization matrix. If $U_n = 1$, at least one microservice instance is allocated on node n.
$U^{u,a,id}_{s,\beta_i}(n)$	The user association matrix. If $U^{u,a}_{id,s,\beta_i}(n) = 1$, the UG u associates with the replica β_i of microservice s allocated on node n for the application a with SFC ID id.
$\underline{U^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1,n_2)}$	The SFC matrix. If $U^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1,n_2) = 1$, the UG u traffic traverses the path from the replica β_i of microservice s_i allocated on node n_1 to the replica β_j of microservice s_j allocated on node n_2 for the application a with SFC ID id.
$\underline{F^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1,n_2)}$	The user flow matrix contains the amount of bandwidth (in Mbps) reserved for the UG u for the communication between the replica β_i of microservice s_i allocated on node n_1 and the replica β_j of microservice s_j allocated on node n_2 to access application a with SFC ID id.
$\underline{T^{a,id}_{s,\beta_i}(n)}$	The computing matrix indicates the execution time (in ms) of the replica β_i of microservice s executed on node n for the application a with SFC ID id.
$\underline{T_{u,s}}$	The user computing matrix indicates the execution time (in ms) of microservice s affecting UG u.
$\underline{T_u}$	The computing latency matrix indicates the total execution time (in ms) of all micro-services associated with UG u.
$\underline{\zeta_U}$	The scene latency matrix indicates the propagation time (in ms) from the VR scene to reach the first microservice in the SFC assigned to UG u.
ψ_u	The user-perceived latency matrix indicates the transmission time (in ms) of a request from the UG u to reach the last microservice in the assigned service chain.
$\underline{\Lambda_u}$	The SFC latency matrix indicates the transmission time (in ms) of requests coming from UG u to traverse the assigned service chain.
$\underline{\epsilon_u}$	The E2E latency matrix indicates the time (in ms) from a VR scene up to UG u after traversing the assigned service chain.

The model decomposes an application A in a set of different microservices S. A specific SFC Identifier (ID) id is associated with each application A. The maximum number of instances per service chain for all microservices is given by β. The replication factor for a microservice s from application a with the SFC ID id is given by $\beta_{a,id,s}$. The MILP model determines the number of instances for each microservice depending on the considered objective (e.g. minimizing deployment costs, reducing E2E latency). Users are aggregated into UGs $u \, \varepsilon \, U$ to access a particular application a and the corresponding microservices. A VR scene vr (i.e. content) is at a given location l in the network area. The fog-cloud infrastructure is deployed across all locations L, each corresponding to a particular Point of Presence (PoP). Each PoP provides computing resources based on the set of nodes N, enabling the deployment of microservice instances based on their requirements and subject to multiple constraints. Each microservice s has a minimum CPU and memory requirement represented by ω_s (in cpu) and γ_s (in Mi) respectively. For instance, a CPU requirement equal to 1.0 cpu (i.e. 1000 millicpu) indicates that each microservice instance requires at least a core to operate properly. Also, each microservice s has a minimum bandwidth requirement represented by δ_s (in Mbps), while ϖ_n corresponds to the associated node cost (e.g. edge and fog nodes have a lower cost than cloud nodes). Node costs are based on Amazon EC2 On-Demand Pricing [18] further detailed in Section V. A binary placement matrix P represents microservice deployments. If $P^{a,id}_{s,\beta_i}(n) = 1$, the replica β_i of microservice s is allocated on node n for the application a with SFC ID id. A user association matrix U represents user chain associations. If $U^{u,a,id}_{s,\beta_i}(n) = 1$, the UG u is connected to the replica β_i of microservice s provisioned on node n. In addition, the SFC matrix U represents specific chain paths assigned to a given UG u. The MILP model determines a specific route for the traffic of each UG u depending on the deployed microservices and in the considered objective.

Multiple decision variables related to computing time T have also been added to the model. $T^{a,id}_{s,\beta_i}(n)$ indicates the execution time (in ms) of a particular replica, while $T_{u,s}$ indicates the execution time (in ms) of the instance of microservice s associated with UG u. The execution time of each microservice instance increases depending on the number of UGs associated. Further details on how execution time is affected by the number of user associations are given in the next section. Also, several decision variables have been included in the model to formulate the E2E latency expected by each UG. First, ζ_U corresponds to the VR scene latency (content location). Second, Λ_u relates to the network latency (SFC) associated with UG u. Third, T_u corresponds to the microservice execution time associated with UG u, while ψ_u relates to the user-perceived latency (user location). Finally, ϵ_u indicates the transmission time (in ms) of the complete E2E path, from a VR scene up to the UG u after traversing the assigned service chain. Objectives and constraints are detailed in the next section, including further explanations on all latency variables. Constraints from [9] have been considered in this extended model. To avoid repetition, only constraints related to novel variables are described.

B. Objectives & Constraints

1) Minimization of the E2E Latency (MIN E2E): This objective relates to the E2E latency reduction of each UG. Multiple constraints have been added to reflect the extensions regarding service chain path selection and microservice execution time. The MILP model decides on which nodes microservices instances should be allocated but also on the number of instances required to minimize E2E latency for all UGs. It could be more beneficial to allocate extra microservice instances to reduce the E2E latency expected by each UG than merely just associating several UGs to the same instance.

Firstly, the constraint shown in (1) states that all deployed microservice instances require at least one UG associated. Secondly, specific chain paths are determined for each UG depending on the deployed microservice instances and the proper chain operation, expressed by the Flow Factor $\Upsilon s_i, s_j$ as shown in (2). Constraint (3) guarantees that all paths are calculated based on user associations and constraint (4) ensures all required paths are associated with each UG.

$$\forall a \varepsilon A, id \varepsilon ID, s \varepsilon S, \beta_i \varepsilon \beta, n \varepsilon N : \sum_{u \varepsilon U} U^{u,a,id}_{s,\beta_i}(n) \geq 1.0 \quad \textbf{if } P^{a,id}_{s,\beta_i}(n) = 1 \tag{1}$$

$$\Upsilon s_i, s_j = I_{a,s_i} \times I_{a,s_j} \times \alpha_{s_i,s_j} \tag{2}$$

$$\forall u \varepsilon U, \forall a \varepsilon A, id \varepsilon ID, s_i \varepsilon S, \beta_i \varepsilon \beta, s_j \varepsilon S, \beta_j \varepsilon \beta, \forall n_1 \varepsilon N, \forall n_2 \varepsilon N : U^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1, n_2) = 0 \quad \textbf{if } U^{u,a,id}_{s_i,\beta_i}(n_1) = 0 \vee U^{u,a,id}_{s_j,\beta_j}(n_2) = 0 \tag{3}$$

$$\forall u \varepsilon U : \sum_{a \varepsilon A} \sum_{id \varepsilon ID} \sum_{s_i \varepsilon S} \sum_{\beta_i \varepsilon \beta} \sum_{s_j \varepsilon S} \sum_{\beta_j \varepsilon \beta} \sum_{n_1 \varepsilon N} \sum_{n_2 \varepsilon N} U^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1, n_2) \times \Upsilon s_i, s_j = \sum_{s_i \varepsilon S} \sum_{s_j \varepsilon S} \alpha_{s_i,s_j} \tag{4}$$

Thirdly, the flow matrix F has been subjected to various constraints to accurately represent network flows: bandwidth capacity limitations (5) and flow conservation (6).

$$\forall n_1 \varepsilon N, \forall n_2 \varepsilon N : \sum_{u \varepsilon U} \sum_{a \varepsilon A} \sum_{id \varepsilon ID} \sum_{s_i \varepsilon S} \sum_{\beta_i \varepsilon \beta} \sum_{s_j \varepsilon S} \sum_{\beta_j \varepsilon \beta} F^{u,a,id,s_i,\beta_i}_{s_j,\beta_j}(n_1, n_2) \leq B_{n_1,n_2} \tag{5}$$

$$\forall u \varepsilon U, \forall a \varepsilon A, id \varepsilon ID, s_i \varepsilon S, \beta_i \varepsilon \beta, s_j \varepsilon S, \beta_j \varepsilon \beta, \forall n_1 \varepsilon N, \forall n_2 \varepsilon N : F^{u,a,id,s_i,\beta_i}_{s_j,\beta_j}(n_1, n_2) = \begin{cases} C_{s_i,s_j} & \text{if } U^{u,a,id,s_i,\beta_i}_{s_j,\beta_j}(n_1, n_2) = 1 \\ 0 & \text{if } U^{u,a,id,s_i,\beta_i}_{s_j,\beta_j}(n_1, n_2) = 0 \end{cases} \tag{6}$$

Fourthly, constraints regarding computing matrices T have also been included. Constraint (7) represents the execution time of each microservice instance affected by the total number of user associations based on its computing cost κ_s. Constraint (8) formulates the execution time of each instance, while constraint (9) determines the computing latency.

$$\forall a \varepsilon A, id \varepsilon ID, s \varepsilon S, \beta_i \varepsilon \beta, n \varepsilon N : T^{a,id}_{s,\beta_i}(n) = \sum_{u \varepsilon U} U^{u,a,id}_{s,\beta_i}(n) \times \kappa_s \quad \text{(in ms)} \tag{7}$$

$$\forall u \varepsilon U, \forall a \varepsilon A, id \varepsilon ID, s \varepsilon S, \beta_i \varepsilon \beta, n \varepsilon N : T_{u,s} = T^{a,id}_{s,\beta_i}(n) \quad \text{(in ms)} \quad \textbf{if } U^{u,a,id}_{s,\beta_i}(n) = 1 \tag{8}$$

$$\forall u \varepsilon U : T_u = \sum_{s \varepsilon S} T_{u,s} \quad \text{(in ms)} \tag{9}$$

Then, several constraints address the novel latency variables. Constraint (10) represents the user-perceived latency. If the instance is deployed on a node far from the UG, the latency is thus higher. Constraint (11) expresses the scene latency and constraint (12) determines the SFC latency. Finally, constraint (13) combines all latency variables, formulating the E2E latency ϵ_u. The E2E latency minimization is then expressed as shown in (14).

$$\forall u \varepsilon U, a \varepsilon A, id \varepsilon ID, s \varepsilon S, \beta_i \varepsilon \beta, n \varepsilon N : \psi_u = \tau_{u,n} \quad \text{(in ms)} \quad \textbf{if } \iota_s \times U^{u,a,id}_{s,\beta_i}(n) = 1 \tag{10}$$

$$\forall vr \varepsilon VR, \forall u \varepsilon U, a \varepsilon A, id \varepsilon ID, s \varepsilon S, \beta_i \varepsilon \beta, n \varepsilon N : \zeta_U = \tau_{vr,n} \quad \text{(in ms)} \quad \textbf{if } \upsilon_s \times U^{u,a,id}_{s,\beta_i}(n) = 1 \tag{11}$$

$$\forall u \varepsilon U : \Lambda_u = \sum_{a \varepsilon A} \sum_{id \varepsilon ID} \sum_{s_i \varepsilon S} \sum_{\beta_i \varepsilon \beta} \sum_{s_j \varepsilon S} \sum_{\beta_j \varepsilon \beta} \sum_{n_1 \varepsilon N} \sum_{n_2 \varepsilon N} \tau_{n_1,n_2} \times U^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1, n_2) \tag{12}$$

$$\forall u \varepsilon U : \underbrace{\epsilon_u}_{\text{E2E latency}} = \underbrace{\zeta_U}_{\text{Scene}} + \underbrace{T_u}_{\text{Computing}} + \underbrace{\Lambda_u}_{\text{SFC}} + \underbrace{\psi_u}_{\text{User}} \tag{13}$$

$$min \sum_{u \varepsilon U} \epsilon_u \tag{14}$$

2) Minimization of the Deployment cost (MIN Cost): The deployment cost estimation has been reformulated in this extended version of the MILP model. This objective relates to the number of active nodes used in the service allocation, resulting in hardware costs and energy consumption. Constraint (15) reflects the relation between the placement matrix P and the node utilization matrix U_n. A node is active only if any microservice instance is deployed on that node. The goal is to reduce deployment costs by selecting nodes with lower cost achieving greener deployment schemes. The minimization of the deployment cost is expressed as shown in (16) by using the node utilization matrix U_n and the node cost ϖ_n.

$$\forall n \varepsilon N : U_n = \begin{cases} 1.0 & \text{if } \sum_{a \varepsilon A} \sum_{id \varepsilon ID} \sum_{s \varepsilon S} \sum_{\beta_i \varepsilon \beta} P^{a,id}_{s,\beta_i}(n) \geq 1 \\ 0.0 & \text{if } \sum_{a \varepsilon A} \sum_{id \varepsilon ID} \sum_{s \varepsilon S} \sum_{\beta_i \varepsilon \beta} P^{a,id}_{s,\beta_i}(n) = 0 \end{cases} \tag{15}$$

$$min \sum_{n \, \varepsilon \, N} \varpi_n \times U_n \tag{16}$$

IV. Virtual Reality (VR) use cases

Fig. 1 illustrates the container-based VR use cases, while Table III shows the correspondent deployment requirements. First, a live scenario has been assessed, in which all services represent a different network function required to capture the VR content. Second, a view prediction service is added to the service chain. The purpose of deploying view prediction services is that only a limited part of the video (i.e. the viewport) is watched by the user. Recently, adaptive tile-based video streaming techniques [19] have been proposed to cut the video into temporal segments and spatial tiles instead of sending the whole content directly to the user aiming to save bandwidth by predicting the user field of view (FoV). Each tile (i.e. video portion) can be requested at a different quality level, prioritizing content within the viewport. Effectively predicting the future user FoV (a few seconds ahead) helps save bandwidth while minimizing video freezing under bandwidth variations [19]. View prediction allows for prefetching ahead of time, so the content is not served from the central server anymore, but from the cluster node where the view prediction service is deployed. The E2E latency formulation has been adapted for this scenario since the user-perceived latency starts in the view prediction service. The model has been adapted as follows:

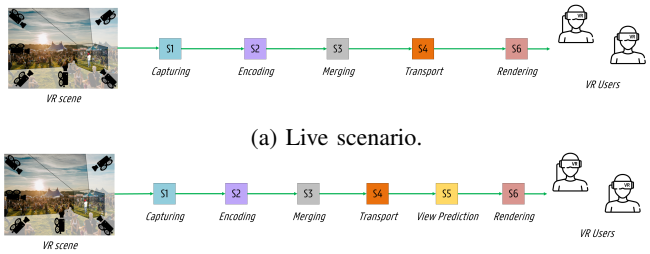
- User computing $T_{u,s}$ (17).
- SFC latency Λ_u (18).
- E2E Latency ϵ_u (19).

The input variable μ_s indicates which microservices are considered when a view prediction service is deployed. The aim is to quantify the differences regarding service placement when view prediction is added to the chain by evaluating both live scenarios. Regarding the deployment properties, UGs are associated with microservice instances based on service slots ρ_s and user costs λ_u. For instance, for the *Capturing* (s_1) service with ten slots, since each UG has a unit cost (i.e. 1), each instance can host up to ten UGs. In addition, the computing cost of each microservice (κ_s) affects the total processing time as previously described.

$$\begin{aligned} &\forall u \, \varepsilon \, U, \forall a \, \varepsilon \, A, id \, \varepsilon \, ID, s \, \varepsilon \, S, \beta_i \, \varepsilon \, \beta, n \, \varepsilon \, N : \\ &T_{u,s} = T^{a,id}_{s,\beta_i}(n) \times \mu_s \quad \text{(in ms)} \quad \textbf{if } U^{u,a,id}_{s,\beta_i}(n) = 1 \end{aligned} \tag{17}$$

$$\forall u \, \varepsilon \, U : \Lambda_u = \sum_{a \, \varepsilon \, A} \sum_{id \, \varepsilon \, ID} \sum_{s_i \, \varepsilon \, S} \sum_{\beta_i \, \varepsilon \, \beta} \sum_{s_j \, \varepsilon \, S} \sum_{\beta_j \, \varepsilon \, \beta} \sum_{n_1 \, \varepsilon \, N} \sum_{n_2 \, \varepsilon \, N} \tau_{n_1,n_2} \times U^{u,a,id,s_i}_{\beta_i,s_j,\beta_j}(n_1,n_2) \times \mu_{s_1} \times \mu_{s_2} \tag{18}$$

$$\forall u \, \varepsilon \, U : \underbrace{\epsilon_u}_{\text{E2E latency}} = \underbrace{T_u}_{\text{Computing}} + \underbrace{\Lambda_u}_{\text{SFC}} + \underbrace{\psi_u}_{\text{User}} \tag{19}$$



(a) Live scenario.

(b) Live scenario with view prediction.

Fig. 1: Illustration of the evaluated VR service chains.

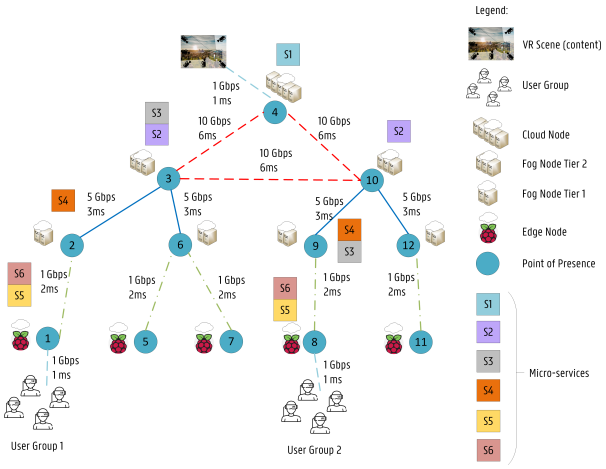


Fig. 2: High-level view of the fog-cloud infrastructure.

V. Evaluation Setup

A. The fog-cloud infrastructure & Input Variables

Fig. 2 illustrates the evaluated infrastructure deployed on twelve locations L (12 PoPs). Table IV shows the hardware configurations of each node. The bandwidth matrix B_{n_1,n_2} is based on the available bandwidth capacity. The maximum replication factor β for each microservice has been set to 4. The VR content has been considered close to the cloud (i.e. location 4), while UGs have been randomly placed on edge locations (i.e. 1, 5, 7, 8, 11). Latency on the access links has been assumed as 1 ms with a connection of 1 Gbps. All latency matrices τ are calculated based on the shown latency values.

B. Optimization Policies

The described MILP formulation has been implemented in Python using the IBM ILOG CPLEX ILP solver [20]. Opposing strategies considering the objectives previously presented have been evaluated: policy A calculates the *MIN E2E* objective while policy B relates to the *MIN Cost* strategy. The model has been executed on the imec Virtual Wall infrastructure [21] at IDLab, Belgium. Cluster nodes have been requested to run the experiments (2 x 8-core Intel Xeon E5-2650v2 @ 2.6 GHz processor with 48 GB of memory). The policies have been executed 30 times and confidence intervals of 95% have been considered.

TABLE III: Deployment properties of the VR use cases.

Application	Service Name	Chain Position	Computing cost (in ms)	CPU (cpu)	RAM (Mi)	Min. Band. (Mbps)	Service Slots	User Cost
Live (a_1)	Capturing (s_1)	1	1.0 (κ_{s_1})	1.0 (ω_{s_1})	1.0 (γ_{s_1})	1000.0 (δ_{s_1})	10 (ρ_{s_1})	1.0 (λ_u)
	Encoding (s_2)	2	2.0 (κ_{s_2})	2.0 (ω_{s_2})	2.0 (γ_{s_2})	1000.0 (δ_{s_2})	5 (ρ_{s_2})	
	Merging (s_3)	3	1.0 (κ_{s_3})	1.0 (ω_{s_3})	1.0 (γ_{s_3})	100.0 (δ_{s_3})	5 (ρ_{s_3})	
	Transport (s_4)	4	1.0 (κ_{s_4})	1.0 (ω_{s_4})	1.0 (γ_{s_4})	100.0 (δ_{s_4})	10 (ρ_{s_4})	
	Rendering (s_6)	5	1.0 (κ_{s_6})	1.0 (ω_{s_6})	1.0 (γ_{s_6})	100.0 (δ_{s_6})	10 (ρ_{s_6})	
Live with View Prediction (a_2)	Capturing (s_1)	1	1.0 (κ_{s_1})	1.0 (ω_{s_1})	1.0 (γ_{s_1})	1000.0 (δ_{s_1})	10 (ρ_{s_1})	1.0 (λ_u)
	Encoding (s_2)	2	2.0 (κ_{s_2})	2.0 (ω_{s_2})	2.0 (γ_{s_2})	1000.0 (δ_{s_2})	5 (ρ_{s_2})	
	Merging (s_3)	3	1.0 (κ_{s_3})	1.0 (ω_{s_3})	1.0 (γ_{s_3})	100.0 (δ_{s_3})	5 (ρ_{s_3})	
	Transport (s_4)	4	1.0 (κ_{s_4})	1.0 (ω_{s_4})	1.0 (γ_{s_4})	100.0 (δ_{s_4})	10 (ρ_{s_4})	
	View prediction (s_5)	5	2.0 (κ_{s_5})	2.0 (ω_{s_5})	2.0 (γ_{s_5})	100.0 (δ_{s_5})	5 (ρ_{s_5})	
	Rendering (s_6)	6	1.0 (κ_{s_6})	1.0 (ω_{s_6})	1.0 (γ_{s_6})	20.0 (δ_{s_6})	10 (ρ_{s_6})	

TABLE IV: The hardware configuration of each node based on Amazon EC2 On-Demand Pricing [18].

Node Type	Amazon Image / Cost ($/h)	Cost (ϖ_n in units)	CPU (cpu)	RAM (Mi)	Band. (Gbps)
Cloud	a1.4xlarge (0.408)	8.0	16.0	32.0	40.0
Fog Tier 2	a1.2xlarge (0.204)	4.0	8.0	16.0	10.0
Fog Tier 1	a1.xlarge (0.102)	2.0	4.0	8.0	5.0
Edge	a1.large (0.051)	1.0	2.0	4.0	1.0

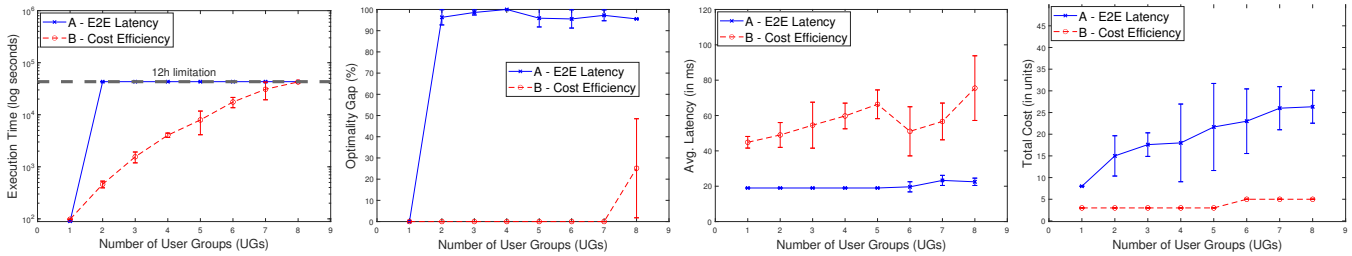
VI. Results

The MILP model shows that optimizing service placement for E2E latency reduction is complex, needing significant execution time (Fig. 3a). A 12-hour limitation has been introduced in the model for all scenarios. Fig. 3b shows the optimality gap retrieved from CPLEX for the live scenario. The optimality gap measures the difference between the best lower and upper bounds. By increasing the number of UGs, the execution time of all objectives increases due to the increased allocation complexity. The MILP model already reaches the 12-hour limitation for two UGs for the *MIN E2E* objective while requiring on average 7.7 minutes for the (*MIN Cost* strategy. A trade-off between latency and deployment costs has been obtained (Fig. 3c and Fig. 3d). The experiments show that the *MIN E2E* objective can still reduce the latency despite not reaching 0% optimality gap and the lack of progress in the best integer solution [22]. The depth-first search algorithm from CPLEX, where each decision variable is a branch in a search tree, retrieves solutions with the best overall values for all scenarios in 12 hours. The application of discount factors in the *MIN E2E* objective could still reduce its execution time but is out of the scope of this work. The *MIN E2E* objective provides on average 19 ms latency to its users while the *MIN Cost* objective achieves on average latency between 40 and 80 ms throughout the experiment. The live scenario is challenging to implement since a large service chain (i.e. 5 services) needs to be deployed to capture the VR content (s_1, s_2, s_3) and then transport the video feeds to each UG (s_4, s_6). Even the *MIN E2E* objective cannot meet the 20 ms MTP threshold for a high number of UGs (i.e. more than 6). In contrast, Fig. 4 shows that adding view prediction services to the service chain significantly reduces the perceived latency since the user FoV is predicted and the content is effectively served from the edge or fog node instantiating s_5. The *MIN E2E* objective obtains latency between 6 and 12 ms while the *MIN Cost* strategy achieves on average latency between 18 and 40 ms throughout the experiment (Fig. 4a). As expected, lower latency translates into higher deployment costs (Fig. 4b). The *MIN E2E* objective already obtains total deployment costs between 25 and 30 units for only four UGs while the *MIN Cost* strategy achieves allocation schemes with total costs of 4 units.

In summary, several factors affect the latency expected by end users in the E2E path. Our model shows that service placement plays a key role, but also service replication and user path calculations. Adding multiple users to the same microservice instance has a direct impact on the perceived latency since services may take longer to respond. Two opposing strategies have been evaluated for distinct use cases. It is difficult to meet the requirements for live scenarios but the addition of microservices in the chain such as view prediction and prefetching further helps to reduce the expected latency and to reach the 20 ms MTP threshold.

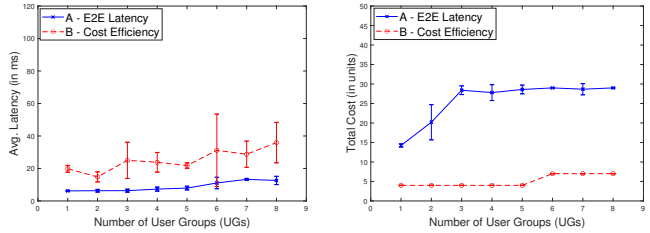
VII. Conclusions

Next-generation VR systems deployed through container-based service chains on fog-cloud infrastructures can meet the maximum 20 ms E2E MTP threshold. Higher latency would lead to poor VR experiences and a lack of adoption for cloud-based VR services. This paper has presented a MILP formulation for the efficient orchestration of VR services leveraging FC and SR. The model optimizes service chain allocations based on E2E latency reduction and applies SR for traffic steering in service chains based on user associations. Opposing strategies have been evaluated for VR use cases showing significant differences. Live scenarios are difficult to implement under limited resources. However, the addition of view prediction services reduces the user-perceived latency. The model shows the benefits of FC and SR concerning E2E latency and provides a reference benchmark for research covering VR service allocation. Results demonstrate that deploying VR components hosted in a fog-cloud infrastructure, combined with view prediction services can support 20 ms latency. As future work, the study of reinforcement learning for resource



(a) Execution Time. (b) Optimality Gap. (c) Latency. (d) Total Cost.

Fig. 3: The evaluation shows that the live scenario is difficult under limited resources. Users experience an average latency of 19 ms even when optimizing E2E latency.



(a) Latency. (b) Total Cost.

Fig. 4: The addition of view prediction to the live scenario helps to reduce the user-perceived latency.

allocation and auto-scaling is already planned to evaluate the performance of these methods at reduced execution times.

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] E. Bastug, M. Bennis, M. Médard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 110–117, 2017.

[2] C.-C. Lee, A. Tabatabai, and K. Tashiro, "Free viewpoint video (FVV) survey and future research direction," *APSIPA Transactions on Signal and Information Processing*, vol. 4, 2015.

[3] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2016, pp. 107–110.

[4] L. P. Berg and J. M. Vance, "Industry use of virtual reality in product design and manufacturing: a survey," *Virtual reality*, vol. 21, no. 1, pp. 1–17, 2017.

[5] 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*, vol. 28, no. 4, pp. 796–826, 2020.

[6] 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.

[7] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards low-latency service delivery in a continuum of virtual resources: State-of-the-art and research directions," *IEEE Communications Surveys & Tutorials*, 2021.

[8] J. Santos, J. van der Hooft, M. T. Vega, T. Wauters, B. Volckaert, and F. De Turck, "SRFog: A flexible architecture for virtual reality content delivery through fog computing and segment routing," in *2021 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2021, pp. 7–12.

[9] 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.

[10] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and Y.-f. Sun, "Qoe-driven joint resource allocation for content delivery in fog computing environment," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[11] G. Rigazzi, J.-P. Kainulainen, C. Turyagyenda, A. Mourad, and J. Ahn, "An edge and fog computing platform for effective deployment of 360 video applications," in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. IEEE, 2019, pp. 1–6.

[12] T. Dang and M. Peng, "Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1594–1607, 2019.

[13] D. You, T. V. Doan, R. Torre, M. Mehrabi, A. Kropp, V. Nguyen, H. Salah, G. T. Nguyen, and F. H. Fitzek, "Fog computing as an enabler for immersive media: Service scenarios and research opportunities," *IEEE Access*, vol. 7, pp. 65 797–65 810, 2019.

[14] D. Alencar, C. Both, R. Antunes, H. Oliveira, E. Cerqueira, and D. Rosário, "Dynamic microservice allocation for virtual reality distribution with qoe support," *IEEE Transactions on Network and Service Management*, 2021.

[15] F. Chiti, R. Fantacci, F. Paganelli, and B. Picano, "Virtual functions placement with time constraints in fog computing: a matching theory perspective," *IEEE Transactions on Network and Service Management*, 2019.

[16] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," *Computer Communications*, 2020.

[17] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, "Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment," *Future Generation Computer Systems*, vol. 104, pp. 131–141, 2020.

[18] A. EC2, "Amazon ec2 on-demand pricing," accessed on 15 March 2021. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

[19] J. van der Hooft, M. T. Vega, S. Petrangeli, T. Wauters, and F. De Turck, "Optimizing adaptive tile-based virtual reality video streaming," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 381–387.

[20] I. ILOG, "Ibm cplex ilog optimization studio," accessed on 15 March 2021. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio

[21] imec iLab.t, "The virtual wall emulation environment," accessed on 15 March 2021. [Online]. Available: https://doc.ilabt.imec.be/ilabt-documentation/index.html

[22] E. Klotz and A. M. Newman, "Practical guidelines for solving difficult mixed integer linear programs," *Surveys in Operations Research and Management Science*, vol. 18, no. 1-2, pp. 18–32, 2013.