SEMANTIC

end-to-end Slicing and data-drivEn autoMAtion of Next generation cellular neTworks with mobIle edge Clouds

Marie Skłodowska-Curie Actions (MSCA)
Innovative Training Networks (ITN)
H2020-MSCA-ITN-2019
861165 - SEMANTIC

WP2 – MEC/RAN integration and MEC-empowered enhancements

D2.2: MEC service-provisioning for the beyond 5G RAN

Contractual Date of Delivery:	M36
Actual Date of Delivery:	31/12/2022
Responsible Beneficiary:	UOA
Contributing beneficiaries:	POLITO, CTTC, UOA, FOG
Security:	Public
Nature:	Report
Version:	V0.1

Document Information

Version Date: 09/11/2022
Total Number of Pages:

Authors

Name	Organization	Email
Dr. Nikos Passas	UOA	passas@di.uoa.gr
Vasilis Gretsistas	UOA	vagretsi@di.uoa.gr
Madhura Adeppady	POLITO	madhura.adeppady@polito.it
Vaishnavi Kasuluru	CTTC	vkasuluru@ctts.es
Klearchos Palias	FOG	klearchos@fogus.gr
Prof. Carla Fabiana Chiasserini	POLITO	carla.chiasserini@polito.it
Dr. Paolo Giaccone	POLITO	paolo.giaccone@polito.it
Dr. Dionysis Xenakis	FOG	dionysis@fogus.gr
Dr. Luis Blanco	CTTC	luis.blanco@cttc.es

Document History

Revision	Date	Modification	Contact Person
V0.1	09/11/2022	Defining a preliminary table of contents	Dr. Nikos Passas

Table of Contents

List of Figures

List of Tables

1 Executive summary

This report includes the SEMANTIC ESR contributions towards the objectives of WP2 (MEC/RAN integration and MEC-empowered enhancements). More specifically, it summarizes the key findings of the ESRs towards Task 2.2 (Prediction tools and strategies for MEC-empowered service provisioning) that includes scientific proposals of the ESRs on MEC architectures, services and ML tools, covering aspects of their integration in the 5G RAN. Architectural and functional upgrades for MEC/RAN integration are also discussed. The document is structured with one section per ESR contributing to WP2, that includes their contribution in the area. A last section at the end concludes the document and sets the target for the final deliverable of WP2, D2.3

2 Microservice Deployment and Placement at Edge Nodes (POLITO)

This section describes our recent work on an interference-aware clustering algorithm for microservice placement algorithm and retention-aware container caching techniques for serverless edge computing.

2.1 iPlace: An interference aware Clustering Algorithm for Microservice Placement

Efficiently deploying microservices (MSs) is critical, especially in data centers at the edge of the network infrastructure where computing resources are precious. Unlike most of the existing approaches, we tackle this issue by accounting for the interference that arises when MSs compete for the same resources and degrades their performance. In particular, we first present some experiments highlighting the impact of interference on the throughput of co-located MSs. Then, we formulate an optimization problem that minimizes the number of used servers while meeting the MSs' performance requirements. In light of the problem complexity, we design a low-complexity heuristic, called iPlace, that clusters together MSs competing for resources as diverse as possible and, hence, interfering as little as possible. Importantly, the choice of clustering MSs allows us to exploit the benefit of parallel MSs deployment, which, as shown by experimental evidence, greatly reduces the deployment time as compared to the sequential approach applied in prior art.

Our numerical results show that iPlace closely matches the optimum and uses 10-63% fewer servers compared to alternative schemes, while proving to be highly scalable.

2.1.1 Introduction

Edge computing is a prominent and promising technology for 5G-and-beyond networks; it enables offloading of service tasks from either mobile devices or the core network to the edge, thus reducing end-to-end latency and resource utilization. While edge computing scenarios vary widely, a prevalent characteristic are services that are composed of simpler components. To witness, service function chains (SFCs) [1] comprise individual virtual network functions (VNFs) or even other, simpler chains; microservice chains are similarly composed of individual microservices (MSs) or other chains. Differences exist: VNFs might run close-to hardware; MSs might run inside general-purpose containers. But while details and terminology certainly differ, many core ideas and issues are very similar across these domains. In the following, for the sake of concreteness, we tackle an MS architecture running inside containers, but we emphasise that our ideas and results apply to SFCs/VNFs as well.

Services as such are deployed by an orchestrator that places, deploys, connects, and configures the needed components in one or several edge data centers, so as to meet the associated Service Level Agreement (SLA). Locally, components run inside a container, facilitated by a hypervisor. Current

hypervisors isolate containers running on the same server by, e.g., placing them on dedicated cores [2], [3], [4], [5], thus allowing to *consolidate* multiple components on the same hardware. Nonetheless, containers still compete for other hardware resources, predominantly memory subsystem resources [6], [7], [8], [9]. Thus, unregulated competition for a server's shared resources by the MSs degrades throughput compared to them running alone on the same server. Such performance degradation experienced by an MS is referred to as *interference* or *noisy neighbor problem [10]*.

Interference is complicated by the multitude of different MSs, each with its own code: they contend for resources differently (e.g., emphasizing memory over I/O) and, hence, experience interference differently [11]. Thus, resources that might have sufficed to meet a particular MS's SLA goal in some combination of components might no longer suffice when combined with other components. This makes guaranteeing SLAs challenging when dynamically consolidating MSs on a limited set of servers, which is the typical operational condition in edge computing.

Recently, several research efforts have been made to address this issue [6], [9], [12], [13]. The proposed solutions use either resource partitioning schemes [9] or supply-demand models [12], [13] to quantify interference. However, none of these methods fully addresses interference completely, as they fail to consider all resources responsible for interference [6]. A few notable approaches [6], [7], [8] have built models to predict the throughput of a target MS when co-located with other MSs and, using such prediction models, they have proposed placement solutions (which component executes where). But even though the prediction models are accurate [6] these placement approaches are somewhat straightforward and suffer from scalability issues.

Unlike prior art, in this paper we propose iPlace, an *Interference-aware Microservice Placement* (IMSP) approach based on clustering and on a prediction model. We start from the observation that, usually, requests for new services do not comprise just a single MS but a set, and also that multiple services might be requested simultaneously. Hence, instead of placing individual MSs sequentially as done in previous work, our solution uses a clustering phase and a placement phase for the sake of scalability. The key idea of clustering the MSs prior to placement stems from the experimental evidence showing that *parallel MSs deployment is much faster than sequential deployment*. Also, while MS clustering has been widely used for placement, e.g., in [16], existing approaches are not suitable for the problem under study as they do not consider interference. Instead, we cluster newly requested MSs so that MSs contending for *different* resources are grouped *together*, thereby *minimizing intra-cluster interference*. Then, we place each cluster in the server whose MSs interfere least with that cluster's MSs, *minimizing intercluster interference* and also ensuring that no SLAs are violated. To account for interference in the clustering and placement phases, we use the *contentiousness* metric [6], [7], which captures the pressure a MS places on shared resources, and build a prediction model inspired by [6], which takes into account the interference among co-located MSs. To summarize, our main contributions are as follows:

1) We introduce a system model capturing the major characteristics of the network system and the virtualized services; we do so by leveraging both previous work [6] and our own experiments;
2) We formulate an optimization problem for IMSP at the network edge to minimize the number of servers used to place the MSs while minimizing the adverse effects of performance interference;
3) Owing to the problem's NP-hardness, we develop iPlace, a heuristic, interference-aware algorithm for cluster-based MS placement;
4) Through extensive simulations, we demonstrate that iPlace efficiently solves the IMSP problem and outperforms state-of-the-art solutions.

To our knowledge, our work is the first to explore clustering to mitigate the effects of interference during MS placement.

2.1.2 Measuring and Predicting Interference

We start by giving experimental evidence on how throughput degrades due to interference among competing MSs, despite a resource isolation setup in the server. We then show how to build a prediction model for estimating the throughput of competing MSs, which is required to develop an interference-aware MS placement solution. We stress, however, that our solution, introduced in Sec. 2.1.3, can work with any other appropriate interference prediction model.

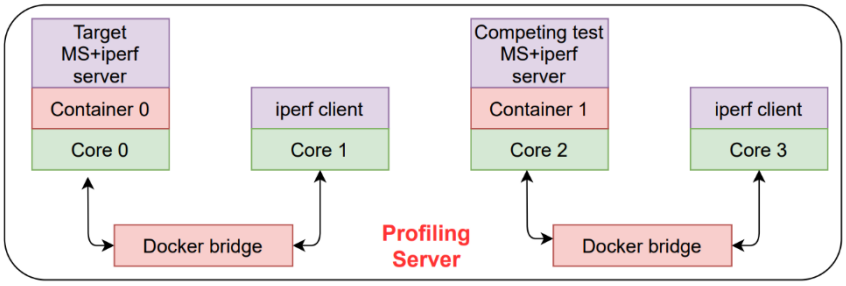


Figure 1. Experimental setting: target MS and competing MS

To observe the throughput degradation due to interference, we carried out several experiments using snort, a well-known open-source intrusion detection system, and pktstat, which displays real-time packet activities. The testbed we used is depicted in Fig. 1. The experiments were conducted on an Intel Core(TM) i7-7700K server with 4 CPU cores, 16GB memory, and 8MB LLC cache shared across all the CPU cores, while individual cores have 1MB L2 cache and 128KB L1 cache (resp.). Each MS runs on a *Docker*

container pinned to a dedicated core using Docker runtime option cpuset-cpus, while iperf3 is used to generate traffic.

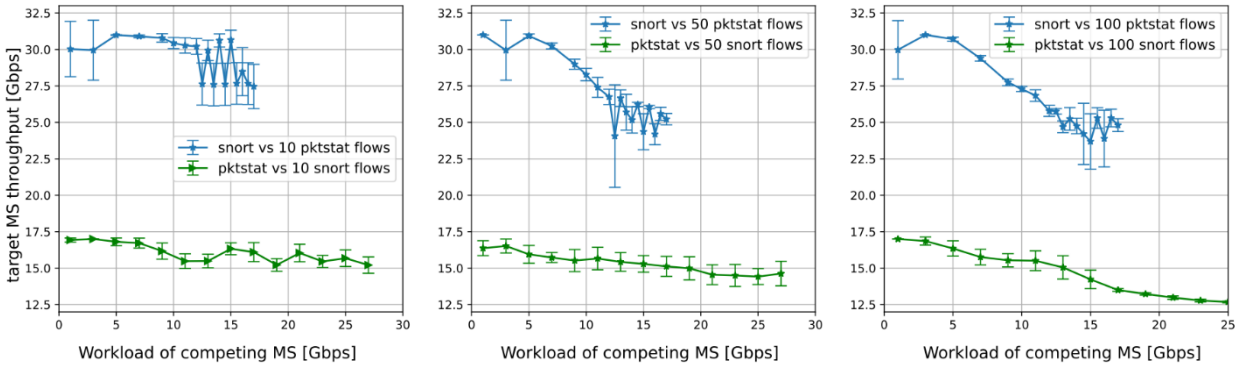


Figure 2. Throughput of the target MS running with a competing MS, as the workload of the latter varies: 10 (left), 50 (center), 100 (right) concurrent flows, and traffic load equally distributed across the flows.

We first fed each MS with 100 flows and ran it individually, on a server dedicated to a single Docker container. Measurements conducted with increasing per-flow traffic load showed that, when running separately, pktstat and snort reach a maximum throughput of 17Gbps and 30Gbps (resp.). We then evaluated the throughput of each MS with the other running as competing MS, for a varying number of flows and per-flow offered load. The results, shown in Fig. 2, highlight a throughput drop of 28.5% for pktstat and of 22.5% for snort, relative to their solo run, both for 100 concurrent flows. Also, as the workload and the number of concurrent flows of the competitor MS increase, the throughput degradation of the target MS becomes more severe. Thus, *despite the isolation of the CPU resources, interference is practically relevant.*

Throughput degradation of the target MS depends upon traffic load and packet processing logic of the competing MSs. As the competitor's number of concurrent flows or its packet rate grow, competition for memory subsystem resources increases, thus degrading performance. It is thus evident that interference plays an important role in MS placement and that inattentive co-location of MSs would seriously degrade throughput.

2.1.2.1 Building a Prediction Model

It is now clear that, to optimally place MSs, it is necessary to predict an MS's throughput taking into account interference. For the sake of concreteness and later evaluation, here we describe the model based on [6], which leverages two main concepts: *contentiousness* and *sensitivity*. Contentiousness measures the pressure (i.e., load) applied on shared server resources by an MS in the presence of competing MSs; sensitivity models the target MS's throughput as a function of its competitors' aggregate contentiousness.

This prediction model includes an *offline profiling phase* and an *online prediction phase*. In the former phase, contentiousness and sensitivity are computed a-priori, considering a target MS running on a server in the presence of a synthetic load. By letting this load increase, the increasing pressure of competing MS(s) on the shared resources is measured. Thus, contentiousness profiling consists of determining a set of vectors, one for each pressure level of the synthetic competitor(s). Profiling sensitivity then builds on a regression model leveraging the throughput of the target MS in the presence of varying synthetic contentiousness vectors. In the *online phase*, the sensitivity model predicts the target MS throughput, given the contentiousness vector of any real competitor(s) as input.

Table 1. Most meaningful system level metrics based on the Correlation Coefficient (CC)

Snort		pktstat	
Metric	CC	Metric	CC
Core-1 EXEC	0.96	System L2MPI	0.98
System READ	0.90	Core-0 IPC	0.98
Core-1 IPC	0.89	Core-1 EXEC	0.96
System WRITE	0.88	Core-2 L2MISS	0.95
Core-2 L3MISS	0.85	System L2MISS	0.95
Core-1 L2MISS	0.83	Core-0 softirqs	0.94

1) Offline Profiling Phase: To evaluate the contentiousness vector, we have considered various system-level metrics (e.g., instructions/cycle, L2/L3 cache misses/hits/occupancy, memory read/write operations) exposed by Intel's PCM framework, which is a performance monitoring API to collect real-time, architecture-specific resource usage metrics. Intel PCM outputs a wide range of metrics but not all of them are relevant. Out of those, we selected components for the contentiousness vector that are highly correlated with the target MS throughput, i.e., the Pearson correlation coefficient is larger than 0.7. Tab. I lists such system-level metrics for snort and pktstat.

The contentiousness vector $V_r^{(k)}(x)$ of MS r depends upon the competing workload, x, generated by k MSs, each with a specific configuration and operational setting (e.g., traffic rate). As in [6], the sensitivity model of MS r, denoted by M_r, is then obtained by training a regression model mapping the contentiousness vector $V_r^{(k)}(\boldsymbol{x})$ into the observed throughput $P_r^{(k)}(x)$. More specifically, we use a Gradient Boosting Regressor model, as sensitivity is a non-linear, non-continuous function of the contentiousness vectors. Further, the experimental results are used to compute the *representative* contentiousness vector $V_r^{(k)}$ of MS r, obtained by *averaging over all* the observed contentiousness vectors with k competing MSs, with respect to the workload values $\boldsymbol{x}$. $V_r^{(k)}$ is then fed as input to the sensitivity model in the online prediction phase.

2) Online Prediction Phase: It leverages the specific contentiousness vectors of the competitors and the sensitivity model of the target MS to predict the throughput of the latter. As an example, let us consider

three MSs, r_a, r_b, and r_c, running on the same server; similar arguments hold for an arbitrary number of MSs. To predict the throughput of r_a in the presence of r_b and r_c, we compute the *aggregate contentiousness* $V^{(2)}_{r_b,r_c}$ jointly imposed by the two competing MSs by combining their representative contentiousness vectors: $V^{(2)}_{r_b,r_c} = V^{(2)}_{r_b} + V^{(2)}_{r_c}$ where, with an abuse of notation, + denotes an appropriate linear operator (e.g., sum for cache occupancy or the cache read/write operations, or average for cache hit or miss probability) applied to each component of the contentiousness vector, to reflect the combined effect of the two competing MSs. We stress that this approach showed to be very accurate [6]. Next, the throughput of r_a, can be predicted via the sensitivity model as: $P_{r_a}(\{r_a, r_b, r_c\}) = M_{r_a}\left(V^{(2)}_{r_b,r_c}\right)$.

Generalizing the above case, the throughput of r_a when running on server s with set $\mathcal{Y}_s \backslash \mathrm{r_a}$ of competing MSs is predicted as:

$$P_{r_a}(\mathcal{Y}_s) = M_{r_a}\left(\sum_{r \in \mathcal{Y}_s \backslash \{r_a\}} V_r^{(|\mathcal{Y}_s|-1)}\right) \quad (1)$$

2.1.3 System Model and Problem Formulation

We now describe the system model under study and formalize the IMSP problem to optimally place MSs.

2.1.3.1 System Model

Let us focus on a single data center and let $\mathcal{S}$ be the set of servers available therein. We consider an online MS placement scenario in which a subset of servers in $\mathcal{S}$ run some pre-existing MSs, each of them currently satisfying its SLA. Let $\mathcal{F}_s$ be the set of pre-existing MSs running on server s; $\mathcal{F}_s = \emptyset$, if s is idle. Then, consider a set $\mathcal{R}$ of requests for MS instances, (possibly) related to different services, arriving at the orchestrator. Let t_r be the minimum required throughput, as per SLA, for an MS $\mathrm{r} \in \mathcal{R}$.

We assume that the data center has ample bijection bandwidth and thus the throughput of an MS depends only upon its server's processing capacity, potentially influenced by interference. Thus, each MS $\mathrm{r} \in \mathcal{R}$ can be placed independently from other $r' \in \mathcal{R}$. Also, each server $\mathrm{s} \in \mathcal{S}$ has CPU and memory resources, denoted by $\widehat{\tau_s}$, $\widehat{\mu_s}$, respectively, while other resource types are sufficiently available. In addition, each MS request r entails CPU and memory demand as denoted by τ_r and μ_r, respectively.

While placing new MSs, pre-existing ones are not moved and their SLAs must be met even after the new MSs have been placed. If an MS request cannot be placed in any of the existing servers without violating the SLAs, then an additional server is provisioned.

Let $y_{r,s} \in \{0,1\}$ with $r \in \mathcal{R}$ and $s \in \mathcal{S}$, be a binary decision variable expressing whether a new MS r should be placed on server s or not, and let $\mathcal{Y}_s = \{r \in \mathcal{R} \,|y_{r,s} = 1\}$ be the set of MSs placed on server s. A server s is active (indicated by $n_s \in \{0,1\}$) if and only if it serves at least one MS r. Using the prediction model introduced in Sec. 2.1.2.1, we can predict the throughput of any MS in a server with co-located MSs. We denote by $P_r(\mathcal{Y}_s)$ the predicted throughput of MS $r \in \mathcal{R}$ when running in server s and competing with MSs in $\mathcal{F}_s \cup \mathcal{Y}_s \setminus \{r\}$, i.e., pre-existing and newly placed MSs.

2.1.3.2 The IMSP Problem Formulation

Given the set of requested MSs, $\mathcal{R}$, the objective is to minimize the number of servers used to place the MSs, i.e.,

$$min \sum_{s \in \mathcal{S}} n_s \tag{2}$$

Subject to system and SLA constraints:

$$\sum_{s \in \mathcal{S}} y_{r,s} = 1 \quad \forall r \in \mathcal{R} \tag{3}$$

$$n_s \leq \sum_{r \in \mathcal{R}} y_{r,s} + |\mathcal{F}_s| \quad \forall s \in \mathcal{S} \tag{4}$$

$$\sum_{r \in \mathcal{Y}_s} y_{r,s} \cdot \mu_r + \sum_{r \in \mathcal{F}_s} \mu_r \leq n_s \cdot \widehat{\mu_s} \quad \forall s \in \mathcal{S} \tag{5}$$

$$\sum_{r \in \mathcal{Y}_s} y_{r,s} \cdot \tau_r + \sum_{r \in \mathcal{F}_s} \tau_r \leq n_s \cdot \widehat{\tau_s} \quad \forall s \in \mathcal{S} \tag{6}$$

$$P_r(\mathcal{Y}_s \cup \mathcal{F}_s) \geq t_r \quad \forall s \in \mathcal{S}, r \in \mathcal{R} \cup \mathcal{F}_s \tag{7}$$

$$n_s \in \{0,1\} \quad \forall s \in \mathcal{S} \tag{8}$$

$$y_{r,s} \in \{0,1\} \quad \forall s \in \mathcal{S}, r \in \mathcal{R} \tag{9}$$

Eq. (3) specifies that a new MS must be placed on exactly one server. Eq. (4) ensures that a server is turned off if no MS is assigned to it. Eqs. (5)–(6) mandate that the memory and computing resource requirement of all (new and pre-existing) MSs allocated in server s cannot exceed the available server memory or computing capability; they also ensure that server s is active if any MS is assigned to it. Eq. (7) leverages (1) and imposes that the predicted throughput for any new and pre-existing MS must satisfy the throughput

required specified in the SLA, thus the mutual interference across all MSs is acceptable. *Theorem:* The IMSP problem in (2), subject to constraints (3)–(9), is NP hard.

The proof, omitted for brevity, shows that any instance of the bin packing problem, which is NP-hard, can be reduced to a simplified, off-line version of the IMSP in polynomial time. This makes a heuristic approach plausible: iPlace, described in the next section.

2.1.4 iPlace: The Interference-Aware MS Placement

The key idea is to partition the set of new MSs into clusters in which the *MSs contend for different types* of resources. Clustering is motivated by experimental evidence showing that, e.g., deploying a batch of 50 MSs using Docker Swarm orchestrator reduces the deployment time by 46% when compared to sequential deployment. Further, while clustering MSs, we reduce the mutual interference of MSs in the same cluster, which allows them to coexist on the same server.

Then, as depicted in Fig. 3, the algorithm works in two phases: the *clustering phase*, which clusters the new MS placement requests based on their contentiousness, and the *placement phase*, which places each created cluster in the server accounting for the pre-existing MSs in the server.

In the clustering phase, we define the *distance* between any $r_i, r_j \in \mathcal{R}\ (r_i \neq r_j)$ of MS placement requests with the following criterion: *larger distance* between MSs means that their corresponding contentiousness vectors are *more similar* and *compete more for similar resources*. Formally, we write:

$$d(r_i, r_j) = |V_{r_i}^{(1)} - V_{r_j}^{(1)}|_2^{-1} \tag{10}$$

The MSs in $\mathcal{R}$ are initially clustered using the mean-shift clustering [15] technique that automatically discovers the number of clusters and the MSs to be included therein based on the chosen distance metric.

After clustering the MSs in $\mathcal{R}$ into clusters, the placement phase starts and the clusters are put in a queue in a random order, processed until each cluster is assigned to a server. More specifically, we define the distance between the server s running pre-existing MSs in $\mathcal{F}_s$ and the cluster $\mathcal{C}$ as:

$$||V_{w\in\mathcal{F}_s}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)} - V_{w\in\mathcal{C}}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)}||_2^{-1} \tag{11}$$

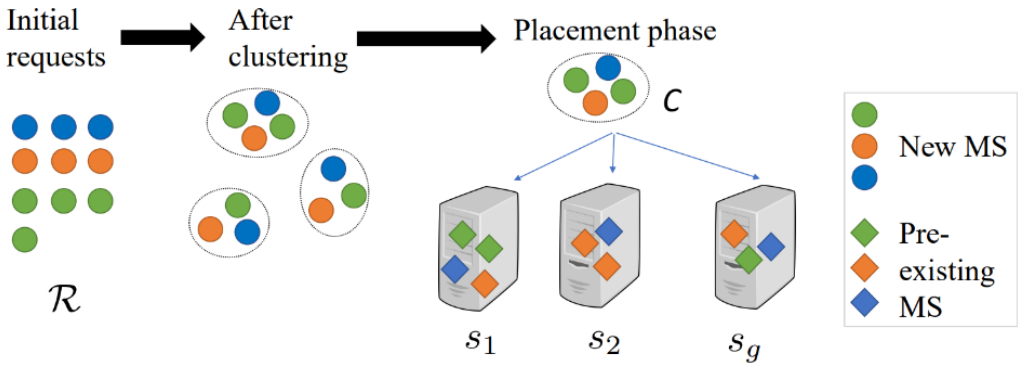


Figure 3. Example of the clustering phase for a batch of requests (color indicates type of resources an MS competes for; shape distinguishes old vs. new MSs).

Algorithm 1 iPlace: Interference-aware MS placement

procedure MsPlacement($\mathcal{R}, \mathcal{V}, \widehat{\mathcal{S}}, \mathcal{M}$)
  $\mathcal{Z} \leftarrow$ MeanShiftClustering $(\mathcal{R}, \mathcal{V})$ ▷ $\widehat{\mathcal{S}}$: set of currently active servers ▷ Initially apply mean-shift clustering on the placement requests based on $\mathcal{V}$
  while $\mathcal{Z} \neq \emptyset$ **do**
    $\mathcal{C} \leftarrow \mathcal{Z}$.pop() ▷ Remove the first cluster from $\mathcal{Z}$ to $\mathcal{C}$
    $\mathcal{A} \leftarrow \{s \in S | \mu_{\mathcal{C}} \leq \hat{\mu}_s \wedge \tau_{\mathcal{C}} \leq \hat{\tau}_s\}$ ▷ Servers with enough resources
    Sort $\mathcal{A}$ in increasing Distance $(\mathcal{C}, \mathcal{F}_a, \mathcal{V})$
    for every s in $\mathcal{A}$ **do** ▷ For each server
      $\hat{r} \leftarrow$ most critical MS in $\mathcal{C} \cup \mathcal{F}_s$
      if PredictSLAViolations$(\hat{r}, \mathcal{F}_s, \mathcal{V}, \mathcal{M})$ = no violations **then**
        $\mathcal{F}_s \leftarrow \mathcal{F}_s \cup \mathcal{C}$ ▷ Place the cluster $\mathcal{C}$ on the server s
        break ▷ Consider a new cluster
    if Cluster $\mathcal{C}$ is not placed **then** ▷ Placing $\mathcal{C}$ in any $s \in \mathcal{A}$ violates SLA
      if $|\mathcal{C}| = 1$ **then** ▷ If size of $\mathcal{C}$ is 1, then a new server is created
        $\widehat{\mathcal{S}} \leftarrow \widehat{\mathcal{S}} \cup \{n\}$ ▷ Start a new server n and place $\mathcal{C}$ there
        $\mathcal{F}_n \leftarrow \mathcal{C}$ ▷ Update pre-existing MSs in n to include cluster $\mathcal{C}$
      else ▷ The cluster is too large and must be split
        $\mathcal{Z} \leftarrow \mathcal{Z}$.append(K-meansClustering$(\mathcal{C}, \mathcal{V})$) ▷ Apply K-means clustering on $\mathcal{C}$ based on $\mathcal{V}$ with $K = 2$
end procedure

where $V_{w \in \mathcal{F}_s}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)}$ is the aggregate contentiousness vector of all MSs placed in s and $V_{w \in \mathcal{C}}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)}$ is the one for $\mathcal{C}$. All the eligible servers having sufficient computing and memory resources to host the MSs of cluster $\mathcal{C}$ are sorted in the increasing order of their distance with cluster $\mathcal{C}$ according to (11). To assess the impact of interference of the new MSs in $\mathcal{C}$ on the nearest server s consisting of $\mathcal{F}_s$ pre-existing MSs, we use the prediction model for the most critical MS placement request $\hat{r} \in \mathcal{C} \cup \mathcal{F}_s$:

$$P_{\hat{r}}(\mathcal{C} \cup \mathcal{F}_s) = M_{\hat{r}}\left(V_{w \in \mathcal{F}_s \cup \mathcal{C} \setminus \{\hat{r}\}}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)}\right) \quad (12)$$

where the most critical MS placement request $\hat{r} \in \mathcal{C} \cup \mathcal{F}_s$ is the one that has minimum throughput drop relative to its solo run, and we considered the aggregate contentiousness vectors of the $(|\mathcal{F}_s| + |\mathcal{C}| - 1)$ competitors of $\hat{r}$ as:

$$V_{w \in \mathcal{F}_s \cup \mathcal{C} \setminus \{\hat{r}\}}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)} = V_{w \in \mathcal{F}_s \setminus \{\hat{r}\}}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)} + V_{w \in \mathcal{C}}^{(|\mathcal{F}_s|+|\mathcal{C}|-1)} \quad (13)$$

If the predicted throughput of $\hat{r}$ following (13) satisfies its SLA requirements, then we can safely place cluster $\mathcal{C}$ on server s. Otherwise $\mathcal{C}$ will be provisionally placed on the next nearest server and the procedure is repeated until we find a server where we can place it without violating the SLA or we ran out of all the active servers. In the latter case, if $\mathcal{C} > 1$, we partition the cluster into two smaller ones using K means clustering with $K = 2$ and add these two new clusters to the end of the cluster queue. If $\mathcal{C} = 1$, we create a new server to place $\mathcal{C}$.

It is worth to note that the algorithm will tend to consolidate the MSs in the minimum number of servers, in line with the considered cost function in (2). The pseudocode of the proposed approach is provided in Alg. 1.

Notably, the overall complexity of the clustering phase is linear in the number of requested MSs; the placement phase is linear in the product of the number of active servers and clusters. Thus, the approach will scale well in the number of allocated MSs; it will *not* grow quadratically in the number of already running MSs. Rather, its worst-case complexity will be $|\hat{\mathcal{S}}||\mathcal{R}|^2$, with $\mathcal{R}$ being the number of newly requested MS instances and $|\hat{\mathcal{S}}|$ the number of currently, still partially empty, active servers.

2.1.5 Performance Evaluation

We evaluate iPlace against the optimum in a small-scale scenario, as well as against state-of-the-art alternatives in a larger scale scenario. To do so, we consider snort and pkstat instance requests, arriving in batches of size $|\mathcal{R}|$, each with its associated SLA, i.e., the required throughput. Specifically, for each MS in $\mathcal{R}$, the latter is chosen from a uniform distribution between 70% and 100% of its solo performance.

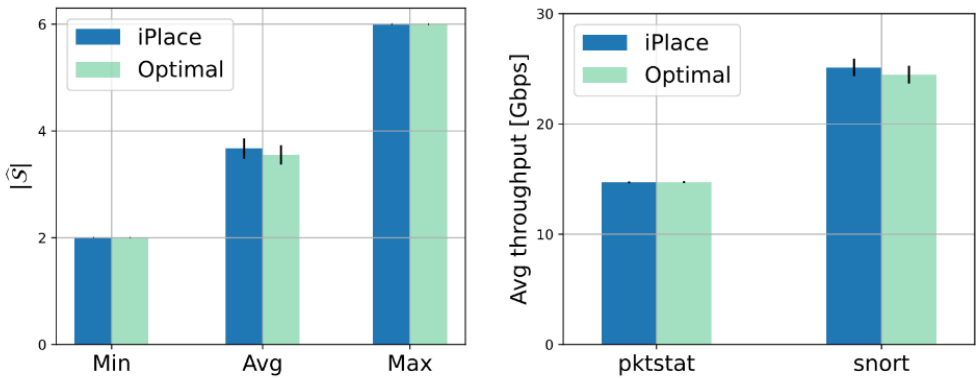


Figure 4. iPlace vs. optimal: number of used servers (left); throughput (right).

Small-scale scenario. We first compare the results yielded by iPlace to the optimal solution. The latter is obtained through brute-force search by generating all possible placement combinations and selecting the one that uses the minimum number of servers while satisfying the SLAs of all MSs. We consider that each request arriving at the orchestrator includes six MSs, and that two servers are already active: one running

a pre-existing snort MS and the other a pre-existing pkstat MS. We repeat the experiment 100 times, each time varying the MSs' required throughput, and compute the confidence interval with a confidence level of 95%.

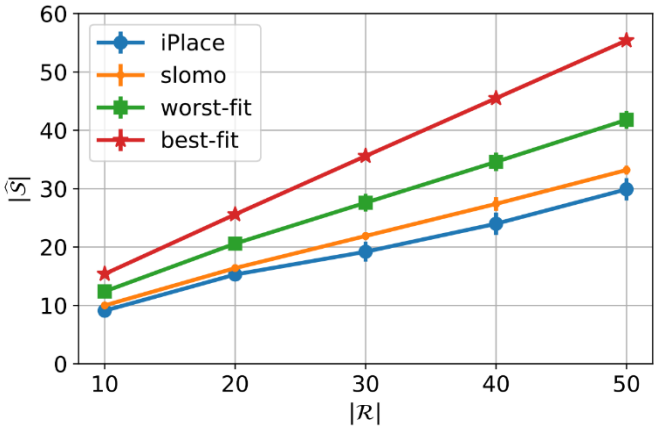


Figure 5. iPlace vs. benchmarks: number of used servers as $|\mathcal{R}|$ varies.

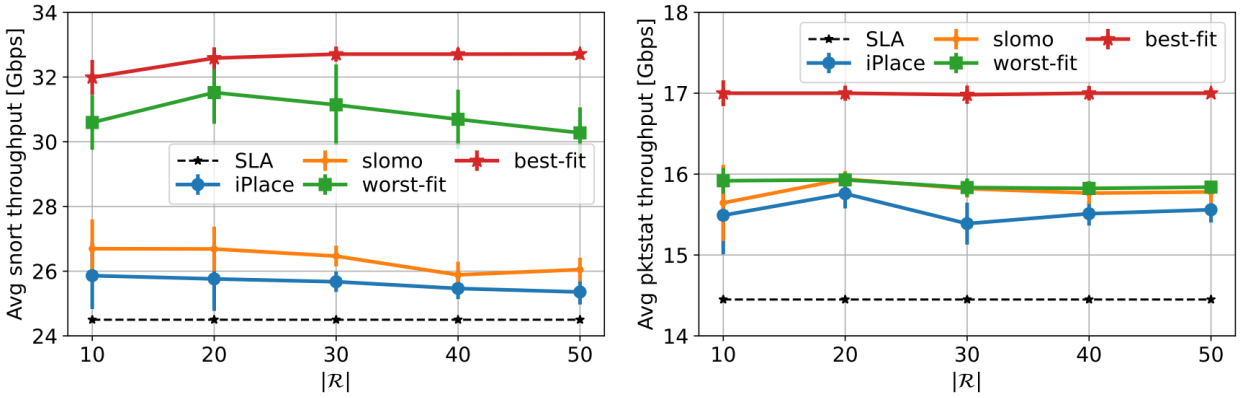


Figure 6. iPlace vs. its benchmarks: average snort throughput (left) and average pktstat throughput (right).

Fig. 4 shows that iPlace requires the same minimum and maximum number of used servers as the optimal, while the average is just slightly higher. Interestingly, Fig. 4 also shows that iPlace can provide better performance, as it reduces the interference among co-located MSs, by placing in the same server MSs competing for different types of resources.

Large-scale scenario. We now vary $|\mathcal{R}|$ from 10 to 50 in steps of 10, such that $\mathcal{R}$ contains an equal number of snort and pktstat instances. The initial number of active servers is fixed to six, each of them running one instance of either pktstat or snort, such that they satisfy their SLAs. As benchmarks, *worst-fit*, *best-fit*, and *slomo* [6] are considered. In worst-fit and best-fit approaches, the new requests are allocated to the server with, respectively, lowest and highest cumulative throughput of the pre-existing MSs running on it. If the request cannot be placed on the server with the lowest (highest) cumulative throughput, a new server is provisioned. In slomo, new requests are placed in a greedy incremental way that evaluates for

every server whether the placement of a new request will lead to SLA violations using the same prediction model as in iPlace. If the request cannot be placed in any of the active servers without violating the SLAs, then a new server is launched.

Fig. 5 shows the number of servers used to place the MSs as $|\mathcal{R}|$, varies. Compared to slomo, worst-fit and best-fit, iPlace utilizes 9.9%, 38.25% and 63.13% fewer number of servers, respectively, for $|\mathcal{R}|$= 50. Importantly, as depicted in Fig. 6, it can do so while satisfying the SLAs of pre-existing as well as newly placed MSs. Clearly, the throughput of pktstat and snort is higher under the alternatives we considered, but they have used a remarkably higher number of servers to place the MSs. Further, we recall that, as shown by experimental evidence, deployment of MS clusters is much faster than sequential deployment. Thus, iPlace can also greatly reduce the MSs deployment time with respect to all its alternatives.

Finally, to demonstrate the scalability of iPlace, we compare the number of calls made to the prediction function, which is the most CPU consuming function, for both iPlace and slomo. To better highlight the significant improvement, we have scaled down the MSs throughput requirements by a factor of 10 so that a higher number of clusters can be placed in the same server. Fig. 7 shows the results obtained using cProfile: for $|\mathcal{R}|$, = 50, iPlace makes 93.58% fewer number of calls to the prediction function than slomo.

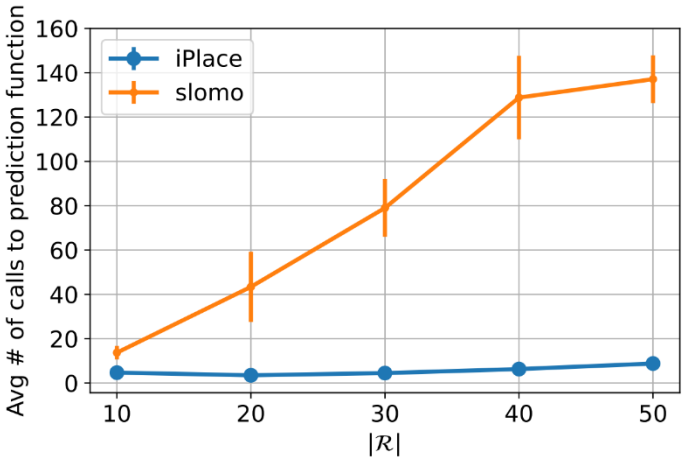


Figure 7. iPlace vs. slomo: number of calls made to the prediction function as $|\mathcal{R}|$ varies

2.1.6 Related Work

The problem of MS, or, alternatively, VNF, placement has been extensively studied in the literature with multiple scopes and objectives. Examples include works that have aimed at properly accounting for user mobility, or at minimizing the delay or the number of used servers. However, most of the existing studies have not considered performance interference, which is instead vital when deciding on which edge server an MS/VNF has to be placed.

Among the few prior approaches that propose interference-aware MS/VNF placement [12], [13], [14] leverage a supply-demand model to quantify interference. In particular, [12] propose an Adaptive Interference Aware (AIA) VNF placement to automatically place VNFs maximizing the total throughput of the accepted placement requests. AIA quantifies interference experienced by the consolidated VNFs through a demand-supply model by profiling each VNF with the aim to measure CPU and memory utilization and meet diverse service requirements. Unlike iPlace, the proposed supply-demand model in AIA does not consider other sources of contention such as LLC, memory bandwidths and software interrupts. Moreover, AIA proposes a sequential deployment of the placement requests to maximize the total throughput of the accepted requests, which incurs a large deployment time.

Many efforts have addressed performance interference using partitioning techniques. ResQ [9] proposes a contention-aware VNF scheduler, based on the observation that LLC and DDIO packet buffers are the critical factors for performance degradation of the consolidated VNFs. This complements our approach, although we did not experience any effect of DDIO since our experiments are always based on virtual interfaces. ResQ combines two techniques: (1) a profiling scheme to determine the LLC partition size required to achieve a target SLA and (2) a Cache Allocation Technology (CAT), which is a hardware-based cache partitioning method to partition the LLC among the competing VNFs. The ResQ scheduler uses both an online greedy heuristic and offline mixed-integer linear programming to find an optimal schedule for the arriving requests based on profiling information. ResQ partitions the LLC among competing VNFs, but it does not isolate other resources degrading throughput, making it an incomplete solution to IMSP. Furthermore, unlike iPlace, ResQ works sequentially on each MS individually and is not able to exploit parallel deployment.

Other work [7], [8] model the contention-induced performance drop suffered by a packet processing unit as a function of competitors' aggregate Cache Access Rate (CAR). Their main observation is that some VNFs are sensitive to co-located VNFs' behavior, and that VNFs can also be ``aggressive'' in causing such sensitive VNFs to suffer. This is coherent with our experimental evidence. Their proposed solution is to co-locate such sensitive VNFs with non-aggressive VNFs. They have considered a single metric to quantify the interference and classify the VNFs. The modern memory architecture is complex; thus, measuring interference based on a single metric may be inefficient. In contrast, the iPlace clustering scheme tends to co-locate the MSs by considering several system-level metrics to quantify the interference.

DeepDive [15] proposes a solution to transparently identify and manage performance interference between co-located virtual machines using a warning system based on low-level metrics and an interference analyzer for determining the culprit resource. Unlike iPlace, which takes interference-aware decisions while deploying the MSs in the first place, DeepDive identifies the interference after the initial placement of the MS and, when necessary, migrates an MS to a different physical machine, which incurs additional migration cost.

The most relevant study to ours is however [6], which presents an interference-aware VNF placement solution. We adopt the same machine learning approach to estimate the interference of slomo. Even if the interference model is the same, the way it is adopted to solve the IMSP is different. Notably, the cluster-based approach in iPlace is compatible with any method to estimate the interference, also different from slomo. To solve the placement problem, slomo proposes a greedy incremental approach to minimize the number of active servers, according to which it verifies whether adding a new VNF request to a server leads to SLA violations for each scheduling request. If there is no feasible solution, slomo provisions a new server to place the request. The greedy incremental approach proposed in slomo is inefficient as it involves tentatively placing newly arrived requests in each active server and checking each VNF individually for SLA violations. Instead, our approach introduces a two-stage method consisting of clustering the MSs that do not contend for the same resources and placing these clusters on separate servers to minimize interference. Moreover, as evident in the experimental results, the batch approach exploited by iPlace dramatically reduces the deployment time compared to slomo.

2.1.7 Conclusions

We have addressed the placement of microservices (MSs) in data centers with the aim to minimize the number of used servers, while meeting the MSs performance requirements. In doing so, we experimentally characterized the gain of parallel versus sequential MSs deployment and the interference among MSs competing for the same resources, and formulated an optimization problem that aims at minimizing the number of used servers. Given the problem NP-hardness, we developed a low-complexity heuristic that aims at placing on the same server batches of MSs that compete for different resources. Our numerical results show that the proposed approach closely matches the optimum and, when compared to existing solutions, reduces the number of used servers by 10-63%, while proving to be highly scalable.

Future work will extend the experimental evaluation to diverse MSs as well as to MSs memory requirements, and it will further enhance the performance prediction model.

2.2 Efficient Container Retention Strategies for Serverless Edge Computing

Serverless edge computing follows an event-driven function execution model in which the function gets executed only when the event triggers, efficiently utilizing the edge server resources. However, starting the containers upon event trigger involves considerable delay known as cold-start latency, hindering the edge services' responsiveness. In a typical serverless platform, cold-start latency can take up to a few seconds, which is unacceptable for latency-sensitive applications. Current literature proposes many solutions to alleviate the cold-start latency by leveraging various container states such as warm containers, pre-warm containers, pause containers, and pre-bake containers. All these approaches have some overhead,

especially in terms of memory and storage, and may not be suitable for all types of applications. Moreover, finding a single solution that reduces the cold-start latency for all applications is challenging. In this work, we focus on characterizing various container states to understand their resource requirement and their impact on reducing the cold-start latency. This is the fundamental step toward designing a container retention strategy to minimize the cold-start latency experienced by the services while using the edge resources efficiently.

2.2.1 Introduction

Serverless computing is an emerging paradigm that has recently witnessed rapid popularity growth. In a typical serverless computing platform, the cloud user develops a cloud function in any of the supported high-level languages and selects the event that should trigger the function execution. The serverless computing platform is responsible for handling everything else, starting from node selection, resource allocation, function deployment, scaling up/down, monitoring, logging, and so on [18]. Unlike cloud computing, where users pay for the resources they reserve, serverless computing uses a fine-grained pay-as-you-use model. Cloud functions are typically executed only when the event gets triggered, reducing unnecessary resource consumption.

In serverless computing, the functions are generally short-lived, with a lifespan of a few seconds [19]. Functions are diverse and may have tight latency requirements. Serverless computing platforms employ virtualization technologies like containers to launch multiple isolated execution environments to serve requests. Each function invocation requires creating a new container with appropriate runtime, which may involve downloading the necessary image from the remote repository and fetching and loading essential libraries and dependencies before executing the actual function. This long delay involved in the initialization setup is known as cold-start latency, and is one of the main performance issues faced by the current serverless computing platforms [19]–[21]. Reducing the cold-start latency is complicated due to the diverse nature of the functions, infrequent function invocations, and unpredictable invocation patterns [18]. To adopt serverless computing widely, it is vital to reduce the cold-start latency while efficiently using the underlying resources.

Although serverless computing platforms were initially developed for web-based microservices and IoT applications, due to their event-driven, elastic, and fine-grained billing process, they have gained popularity in data-intensive applications such as data analytics, streaming, and augmented/virtual reality [21]. In contrast to traditional serverless applications that run just a single function when invoked, complex applications such as above demand application logic spanning multiple functions that share data and intermediate states. Function chains exaggerate the cold-start latency as each function in the chain needs to be created newly whenever the request arrives, known as cascading cold-start.

Current literature proposes many solutions to alleviate the cold-start latency by leveraging various container states such as warm containers [19], [23], pre-warm containers [24], pause containers [21], and pre-bake containers [20]. One of the most widely used techniques to mask the cold-start latency is the warm container or live container. A warm container refers to keeping the container instance alive in the memory after the current function invocation for serving later requests. The cold-start latency is negligible when the warm containers are reused for serving the later request of the same service type. However, a warm container utilizes a certain amount of memory and storage, degrading the server's resource utilization. Pre-warm containers are those containers that already have certain required libraries pre-installed. Generally, serverless platforms like OpenWhisk [25] maintain a pool of pre-warmed containers so that a pre-warmed container can be used directly when the event triggers. The pre-warm containers still suffer from certain start-up latency, as the actual function code still needs to be injected into the containers. The pre-warm containers also occupy memory and storage, but less than that of warm containers. The containers with network stack pre-installed refer to as pause containers. Pause containers do not occupy the memory but will still need storage space. The process of checkpointing a live container and later restoring it whenever the request comes is known as container pre-baking. Pre-baked containers occupy storage. Both pause containers and pre-baked containers still suffer from start-up latency higher than pre-warm and warm containers but less than cold containers.

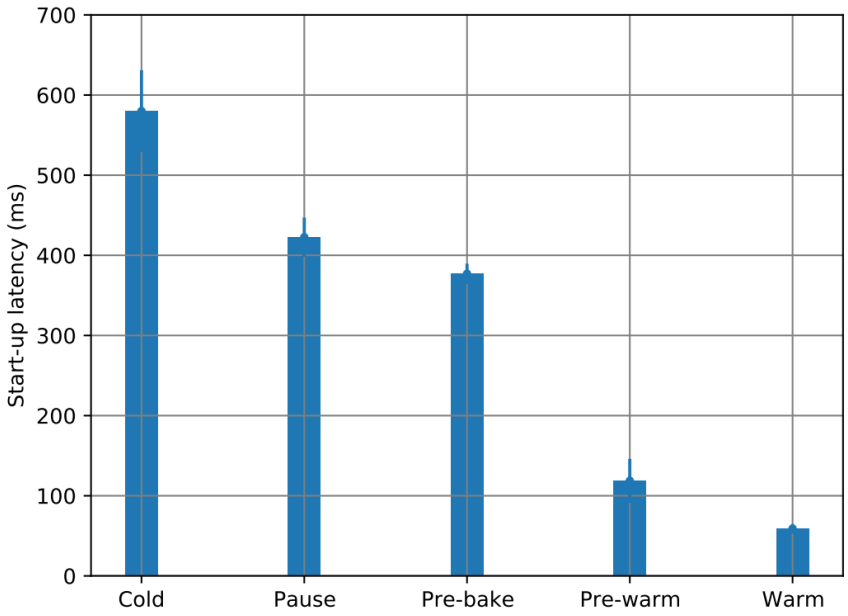


Figure 8. Start-up latencies of various container states

2.2.2 Preliminary Experimental Results

To measure the start-up latency and resource overheads of various container states described above, we conducted experiments using a simple Python function. The experiments on warm, pre-warm, and cold

containers were conducted on OpenWhisk, an open-source serverless platform. OpenWhisk does not support pause and pre-baked containers, so we used the Docker start command to measure their start-up latencies. In particular, we measured the initialization time and time taken to start an already created container with a network stack as start-up time for pause containers. In case of pre-baked containers, the time taken to restore an already checkpointed container snapshot is measured. CRIU tool is used to take the snapshot of the running container. We also measured the memory and storage space needed to retain the considered container states in the server. The experiments are repeated three times, and Fig. 8 shows the average start-up latency of considered container states with confidence intervals for a 95% confidence level. As shown in Fig. 1, the start-up latency of the cold container is the highest, while the warm container has the least start-up latency. All the considered container states have resource overhead in terms of memory and storage, as demonstrated in Table 2. As pre-warm and warm containers are already running in the memory, they consume both memory and storage, while pause and pre-baked containers need just storage space.

Table 2. Resource overheads of various container states

Container state	Memory utilization [%]	Storage [KB]
Cold	0	0
Pause	0	16
Pre-bake	0	937
Pre-warm	0.7	36
Warm	29	40

Thus, keeping the containers in warm/pre-warm/pause or pre-baked states reduces cold-start latency, but they consume nonnegligible storage and (or) memory resources. But if we do not leverage any of these states, there will be high cold-start latency impacting the responsiveness of the offered services. Moreover, the cold-start problem gets exacerbated in serverless edge computing scenarios, where the edge nodes have limited computing, memory, and storage resources. Hence, deciding a container retention time leads to a tradeoff between the average cold-start latency and resource utilization at the serverless edge nodes.

In this work, we investigate resource overheads of various container states and the tradeoff between the average cold-start latency and resource utilization for a typical serverless edge computing scenario. Our main idea is to leverage the characteristics of various container states proposed in the literature and propose an efficient container retention strategy for the edge nodes that minimizes the average cold-start latency while efficiently utilizing the limited edge resources.

3 Streaming Media over HTTP in MEC-empowered Networks (UOA)

In this section, the DASH protocol, related work on the content delivery and placement in MEC-empowered Networks and our proposal in content delivery, are presented.

3.1 Dynamic Adaptive Streaming over HTTP (DASH)

Dynamic adaptive streaming over HTTP (DASH) is the dominant method for delivering video in mobile networks. According to this method the original video target file is transcoded into different video bitrates and it is split into equal time duration segments. The video segments are stored in the content delivery network (CDN) where the end user can switch into different resolutions during the video streaming in a segment-by-segment basis. In pre-5G networks, DASH method had to deal with fluctuations of the wireless channel and the backhaul links. Thus, it effectively served its main scope which is to deliver seamless video content. In 5G/B5G networks where the setup can be more complex, binding together heterogeneous computation, storage, capacity and network resources, traditional DASH lacks the intelligence for fully utilization of MEC and Network slicing (NS) potential.

3.2 Related work

This section includes the content placement and delicery in MEC-empowered Networks related work in different approaches.

3.2.1 Cooperative Content Caching in MEC-Enabled Networks

In [26] they investigate how the content placement can be formulated and optimized under the MEC-enabled Heterogeneous Cellular Networks where the formation of MEC service areas shall enhance the MNOs adaptation to local spatiotemporal file popularity in smaller areas and the actual topology and capabilities of HCN nodes. They provide a 0/1 multiple knapsack (*ZOMKP*) formulation of the cooperative content placement problem within a given MEC cluster with constraints of the heterogeneity of cache sizes of cluster nodes, the no-repetition of popular video files in the same cluster, the non-uniformity of content popularity and the un-coded placement of video files. Focus on the downlink direction of a multi-tier MEC-enabled HCN, where each tier consists of HCN nodes of similar networking capabilities. They consider that the MNO employs a cluster formation algorithm to group a heterogeneous set of nodes that will perform both content placement and delivery. Figure 9 below shows a cache-enabled MEC cluster that can be composed by HCN nodes belonging into different tiers and a content placement and delivery (right).

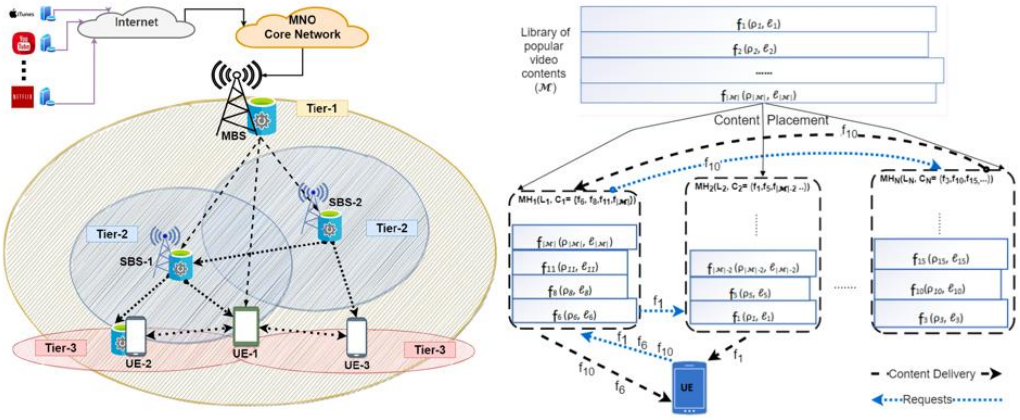


Figure 9. System model of content placement and delivery

In this work, dynamic programing is being used in the proposed Algorithms while Matlab is the software tool for their results. They assess the performance of the proposed content placement strategy and make a comparison between the two widely used strategies: random and greedy content placement strategies.

3.2.2 Quality of Experience in ICN

In [27] they focus in Adaptive streaming with cache partitioning. Their study focuses on the interplay between consumer-side adaptation and in-network cache placement which provide an opportunity to observe adaptation-level dynamics that vary by hop distance (Figure 10). The observations show the need of safe-guarding cache capacity for a particular bitrate to facilitate effective cache placement. Consequently, it leads to the design of the RippleCache, a cache partitioning principle. A RippleCache implementation must identify appropriate caching decision criteria so that placements may form partitions and implement a negotiation mechanism to assure a fair sharing allocations of cache capacity at nodes on the intersecting paths. They also optimize the RippleCache. RippleClassic is an optimization formulated as a binary integer programming (BIP) problem. Its solutions are cache placements for adaptive video content under varied network conditions and preferences. RippleClassic decides content placement among caches, without explicit knowledge of the interplay between caches and consumers. This information is encoded in and modelled by a reward function where its design relies on the following intuition: A cache hit is valuable only if the transfer of video content from that cache to the consumer can be reliably sustained for the requested bitrate.

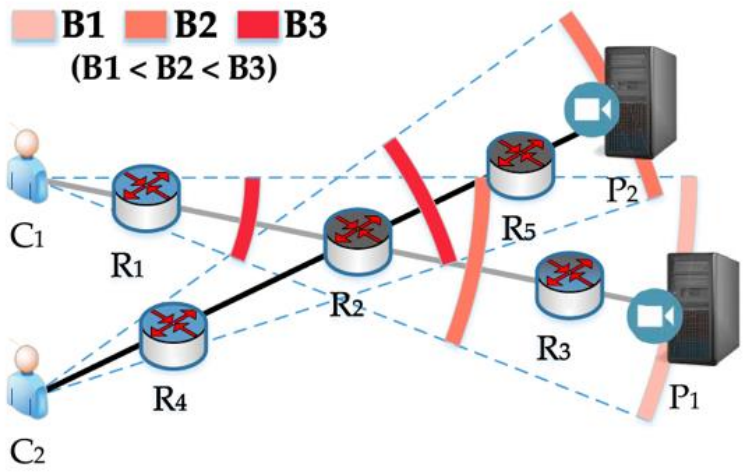


Figure 10. Cache partitioning by encoding bitrates along each forwarding path.

The Gurobi optimizer and BRITE: An approach to universal topology generation are the tools used in this work. Evaluation using standard QoE metrics by DASH Industry Forum. From the standard set, they adopt three metrics, Average Video Quality, Bitrate Switch Count and Rebuffer Percentage.

3.2.3 Optimization of Video Segment Caching

In [28] is shown how caching and transcoding at multi-access edge computing (MEC) server and wireless resource allocation in eNodeB interact with each other and together determine the quality of experience (QoE) of DASH clients. The relationship among the three factors has not been explored, which has led to limited improvement in clients' QoE. They consider a scenario where multiple clients can request video content from an eNodeB (Figure 11). The eNodeB can download the requested video segment from a) the local MEC server, b) the cooperative MEC servers and c) the cloud server. The function of the MEC server consists of video segment caching, transcoding, context awareness and a cooperation mechanism, which can utilize the information of the client's playback status and channel state to make the decision on the video segment caching and transcoding strategy and provide the wireless resource allocation strategy for eNodeB. Refined MEC Caching Mechanism. They select the video segment as the smallest unit in the MEC caching mechanism. In each cache update period, the delete priority of each video segment is calculated as the basis for the video segment delete strategy. The MEC caching mechanism consists of a) MEC caching pre-deployment, b) MEC caching partitioning, c) video segment delete strategy, d) MEC caching space transfer and e) a cooperation mechanism. They treat the problem of Joint optimization of segment caching, transcoding and resource allocation as a high-dimensional nonlinear

mixed-integer optimization problem. Thus, they decompose the problem into multiple subproblems.

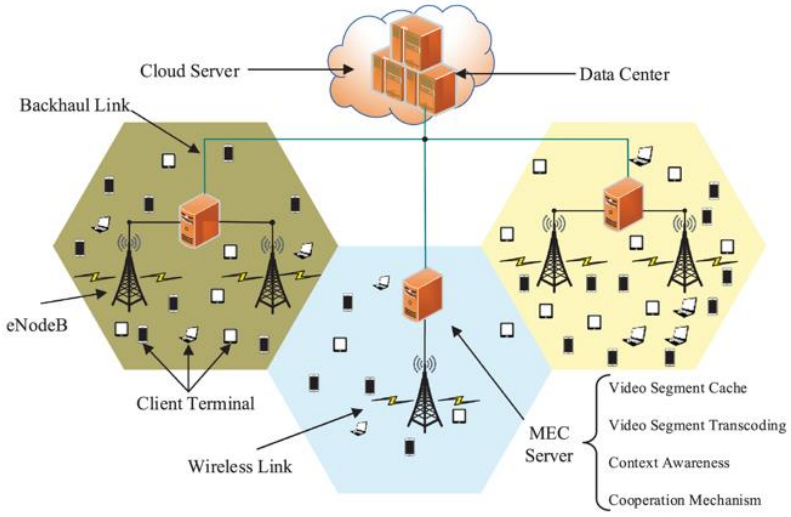


Figure 11. Video Streaming Service in the MEC Assisted Mobile Communication Network.

3.2.4 MEC-Enhanced Video Streaming: Proof-of-Concept

In [29] they aim to the implementation of the proof-of-concept (PoC) for MEC-enhanced mobile video streaming service. They are the first to implement mobile video service enhanced by the MEC concept and architecture. Their model can be deployed based on the RTP/SIP streaming or HTTP adaptive streaming (HAS), allowing to support real-time video quality adaptation. The user app is an android app for the video streaming which is also capable of reporting to the MEC-server about the radio network conditions to effectively support the video quality adaption. When the app is launched, an activity is started that enables the search interface to the user. The end user chooses a video from the search activity to watch and then the play activity will also start to play the target video. The CQI reporting service will be launched at the same time to keep reporting the CQI values to the MEC App during the whole period of the play activity. The MEC app in the MEC-server acts as a helper in mitigating the load on the origin video servers and making best use of the available resources to enhance the user QoE by minimizing the perceived latency. The MEC app provides two functions to achieve these goals respectively: cache management and video quality adaptation. They present two machine learning models for popular video and CQI value predictions in the MEC app. For the video prediction they use the K-Means clustering algorithm. To avoid a too large or too small cluster which would lead to poor performance, Canopy clustering algorithm is applied to get a suitable number of clusters before the K-Means algorithm starts. The integration of these two algorithms leads to better prediction accuracy and the model can be described in the following 3 steps: a) Collecting the training dataset, b) Determining the cluster number and c) Determining the respective cluster for each video. For the CQI Value

Prediction, in Figure 12, they have pointed out that that due to the considerable RTT between the mobile device and the remote video content server, the mobile device radio network conditions are significantly changed. This issue deepens the difficulty of video quality adaptation at the MEC app. The develop a machine learning model for the MEC app to predict the new CQI values of the mobile device during the RTT. To predict these values, they employ the random forest algorithm. They aim to learn the pattern between the known CQI values and the video quality class after RTT. They perform their algorithm is two steps: first the MEC app records the known CQI values and at a second stage they train the machine learning model with the random forest algorithm. They quantify the performance of the cache replacement strategies while apply the prediction model, they use Hit Ratio. Higher Hit Ratio means that the more frequently requested videos are cached in the MEC server. Consequently, the cache replacement strategy can effectively reduce the streaming latency. They evaluate the CQI value prediction by splitting the data set into two different groups where the first and bigger is used for training model and the second to examine the ratio R_c of the correctly classified and the ratio $1 - R_c$ of the incorrectly classified. Higher R_c denotes better MEC app ability to deal with RRT time and make more accurate predictions.

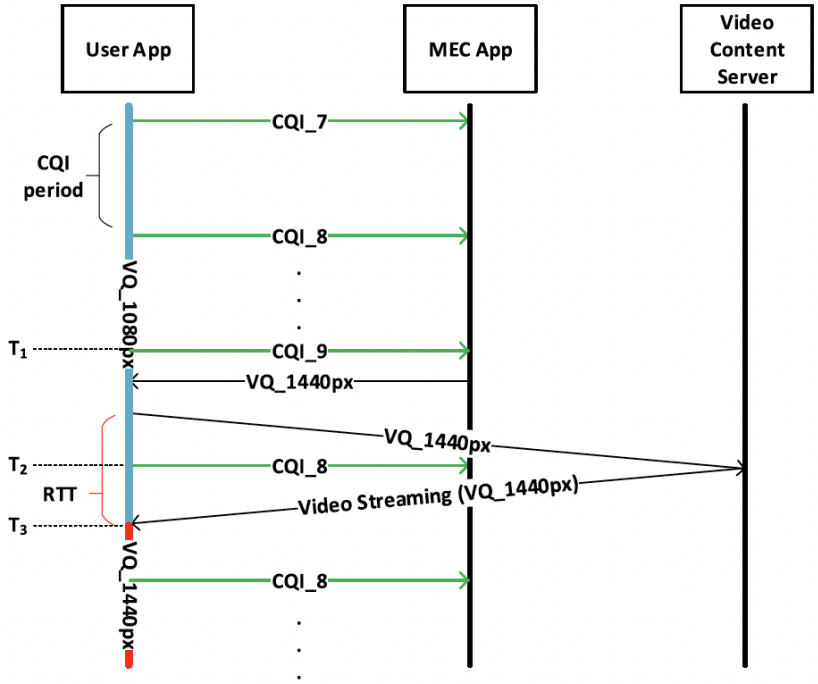


Figure 12. The influence of the RTT on the video quality adaptation

3.2.5 Edge Computing Assisted Streaming

In [30] an optimized solution for network assisted adaptation specifically targeted to mobile streaming in MEC-enabled environments, is presented. They have proposed a heuristic algorithm with minimum need for parameter tuning and low complexity. The purpose of this work is to show the efficiency of their proposed solution and quantify the privileges of the network-assisted adaptation over the client-based approaches in the mobile edge computing scenarios. In this work, they try to solve the problem of optimization in the proposed network assisted bitrate adaptation solution. Also, they aim to investigate how, and at which cost the QoE and fairness between mobile video streaming clients could be improved through network assisted adaptation compared to client-based approaches. In this multi-access edge computing DASH system, the coordinators (Figure 13) perform periodically two functions. First, client-to-edge-server mapping and second, per-client video bitrate selection. Thanks to utilization of the MEC-empowered system features, these functions are done with the help of radio access level information received from the base stations. Application-level information from the mobile streaming clients is received too. The goal of the client-to-server mapping is to act as a load balancer. The per-client video bitrate selection is performed periodically and as optimally as possible. Optimal bitrate selection in their case, means maximizing QoE and including a fairness factor too.

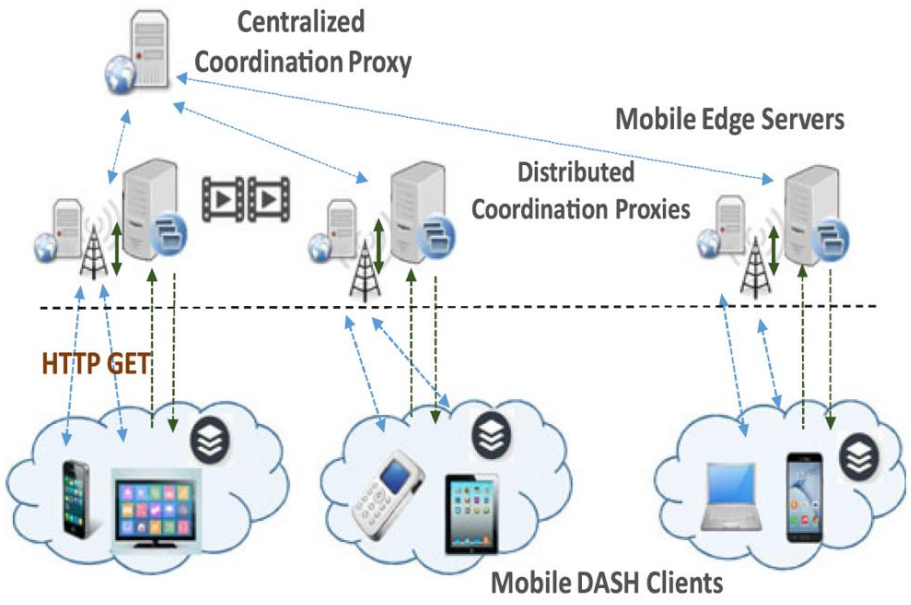


Figure 13: Multi-access edge computing (MEC) assisted DASH system for the large scale of mobile clients

In their system model, the coordinators handle the clients to server mapping and the bitrate adaptation process using the data from the associated base stations and the connecting mobile clients. In the first case, the local coordinator close to the edge servers, send their data to central coordinator for the selection

decisions while the bitrate adaptation is performed by the individual coordinators. Their system uses the second case where the coordinators perform both functions of clients to server and bitrate adaptability which facilitates the decentralized implementation of the network-assisted optimization solution in their system design. The joint optimization for each client is defined as a utility maximization problem with an integer non-linear programming formulation using Greedy MSMC Algorithm to solve it.

3.3 Our Proposal

Different from the existing works, we aim to implement a system for video streaming over HTTP where there is more than one single MEC-enabled proxy server. The goal of this system is to avoid stalling which is mainly caused by cache miss**.** The proposed system model is consisted of three network parts (Figure 13).

3.3.1 System Model

There is a user **u** and a CDN, where a http server is located and serves as ground truth. In this server is stored a file library with an amount of files **F**= {f_1, f_2, f_3, f_M,}. Each file $\mathbf{f_m}$ has **Q** different qualities **q**= {1,2,3…Q} and is partitioned in **S** segments s={1,2,3…S} of equal duration. In the middle and between the user u and the CDN, there are HTTP proxies with a cache size $\mathbf{L_p}$ where **p** is the identifier of the proxy. These proxy-servers can be a BS or a MEC-server which is collocated with the BS. Network Slicing assures reserved resources in both user-to-BS and BS-to-CDN links which guarantee an average throughput of $R_{1,p}$ (user-to-BS link) and $R_{2,p}$ (BS-to-CDN) where p is the proxy identifier. This system model aims to run an adaptive video bitrate selection algorithm on the user application layer to solve a selection problem. We consider as known the i) Cached segments in the MEC-enabled proxies' caches with capacity L_i, ii) Reserved resources

and iii) Average throughput R_1, R_2, the scheduler decides which proxy will deliver every single segment.

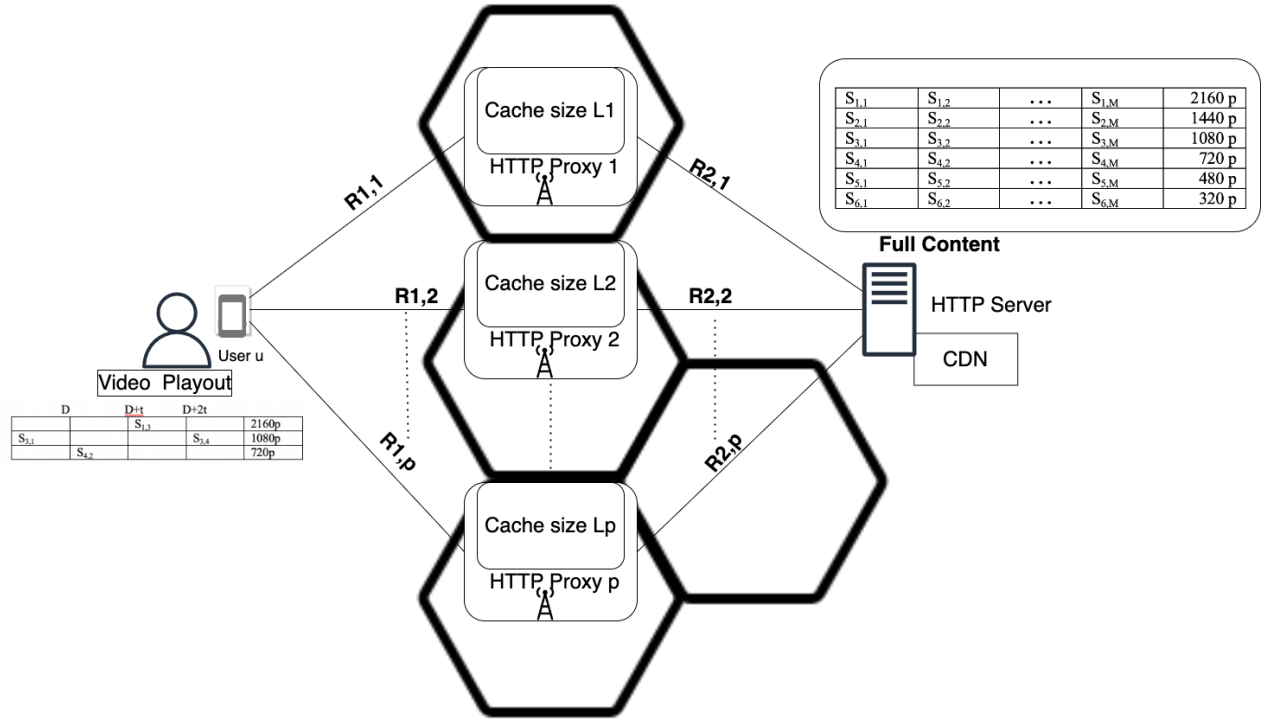


Figure 14. Proposed System model

We focus on a video service request sent (Figure 14) by a mobile video user u to a set of **P** MEC-empowered proxies **p = {1,..P}**. We consider that the video service request concerns a tagged video file **f**, which is available in **Q** (**q={1,2,…Q}**) different video bitrates at the CDN and it is partitioned into **S** (s={1,2,…S}) video segments of equal duration $\mathbf{t_s}$. For each proxy we will use a matrix B_p = QxS, where there are 1s for cache hit and 0s for cache miss. For instance, if the 3rd segment (s_3) is cached in the 4th (q_4= 1080p) resolution then $B_{4,3}$= 1. In the other case (cache miss), it will be equal to 0.

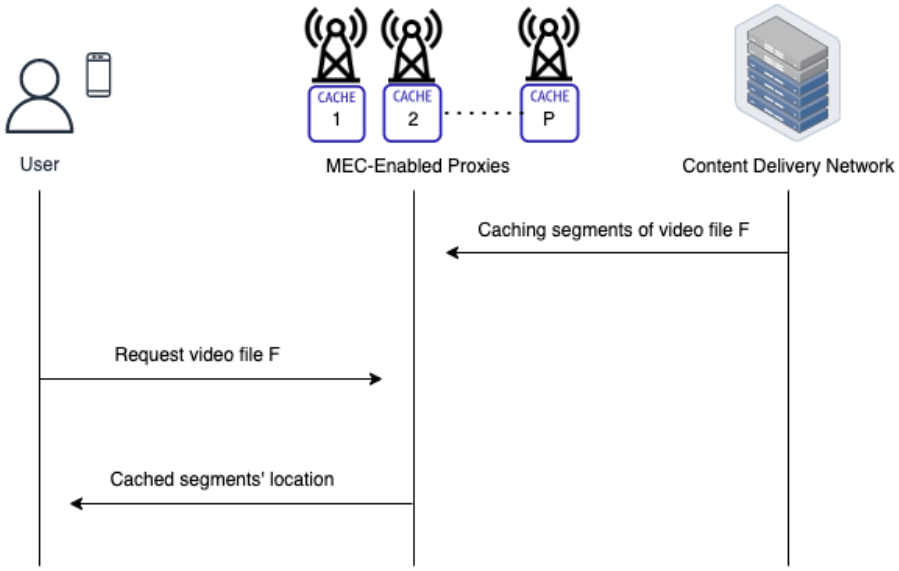


Figure 15: Signaling flow for user video file requests

3.3.2 Problem Formulation

MEC-enabled proxies delivery sequence as a matrix M = 1xS , the value of each cell is the proxy identifier p, which delivers the specific segment, for a target video file f, video bitrate q ∈ {q_{min},… Q} and a user u.

Input: [u, q, Q, S, V, {t_s}, {|$f_{q,s}$|}, $X_{p,f}$, $C_{1,p}$, $C_{2,p}$, D]

$$M = [m_1 .. m_S], m \in p = \{1, .. P\} \quad (1)$$

w.r.t.

$$\frac{X_{p,f}(q,1) * |f_{q,1}|}{D} \leq R_{1,p} \quad (2)$$

$$\frac{\overline{X_{p,f}(q,1)} * |f_{q,1}|}{D} \leq R_{2,p} \quad (3)$$

$$\frac{\sum_{s=2}^{s'} |f_{q,s}|}{\sum_{s=1}^{s'-1} t_s} \leq R_{1,p}, \forall\, s' : 2 \leq s' \leq S \quad (4)$$

$$\frac{\sum_{s=2}^{s'} \overline{X_{p,f}(q,1)} * |f_{q,s}|}{\sum_{s=1}^{s'-1} t_s} \leq \min(R_{1,p}, R_{2,p}), \forall\, s' : 2 \leq s' \leq S \quad (5)$$

$$R_{1,p} \leq C_{1,p}, R_{2,p} \leq C_{2,p}. \quad (6)$$

Denoting the complementary of the binary variable x ∈ {0,1} as $\bar{x}$, equations (2) and (3) assuring that the initial playback delay will not exceed D seconds when the first segment is cached or not cached respectively for both cases. In (4) we make sure that the segments upon the delivery of the first one, will be delivered

in time through the BS-to-user link (R_1), aiming to absolutely mitigate video stalling. The equations in (5) safeguard the timely delivery of non-cached video segments $s \geq 2$ through both the R_1 and R_2 links (end-to-end delivery). Eq. (6) constrains the $R_{1,p}$ and $R_{2,p}$ not to exceed the prescribed capacity thresholds $C_{1,p}$ and $C_{2,p}$.

3.4 Conclusions

In this section the DASH protocol was described and the related work in MEC-empowered video streaming was presented. Moreover, we proposed a system model and formulated a selection problem for the emerging scenario of multi-proxy MEC-enabled for video streaming. We aim to build an algorithm which solves the abovementioned problem and as a future step, to work on the caching strategies for the same scenario of multi-proxy network environment to mitigate video stalling at the end-user side.

4 Resource Orchestration For Massive Connectivity At The Network Edge (FOGUS)

The 4th Section of this document presents our latest work in detail, with respect to the Resource Orchestration techniques, at the Network Edge. To this extent our work has been divided into the following Sections: Section 4.1 - Introduction, Section 4.2 - State Of The Art, Section 4.3 - System Model, and finally Section4.4-Algorithms.

4.1 Introduction

In recent years, the rapid developments of the technology has provided us with a big variety of portable devices such as smartphones and other smart wearables (i.e smart watches, smart glasses, etc). As an outcome, countless applications have been developed over the years offering useful content to the user, but in order to provide personalised or special content sometimes it is necessary for an application to have access to some of the user's personal information, such as his location, his contacts list, etc. At the time being, there is a wide range of mobile applications serving such purposes and providing relative content for example, information about the news, the weather, recommended advertisements and many more. In order to have such dynamic content on our personal devices, we must provide our application with the respective permissions. In this case, this is usually done through the GPS *(i.e Global Positioning System)* sensor, which is supported almost for every mobile device existing today out in the market. In the paragraph following next, we describe how the GPS works in that case. The GPS sensor locates a specific position on a map, from signals it receives from the satellites of the GPS System. This assumes that the mobile device should receive signals from at least 3 satellites *(i.e minimum possible case)* and this signal has not undergone any severe distortion as it reaches its destination *(i.e user's mobile device)*. Such distortions can be caused by either reflections in the ionosphere *(i.e . the satellite is at a large angle with respect to the device)* or by the in-between existence of various materials that do not allow the signal to pass *(i.e such as buildings that have bricks, metals, glass, etc)*. Therefore, for the best possible reception of the signals and consequently to have better accuracy when measuring a location, it is preferably to be in outdoor locations and away from buildings or other obstacles that could reflect the signals of the satellites. As a consequence, it is very hard and sometimes impossible to use the GPS sensor to determine the exact location of a device when the user is in Indoor locations. To this extent, the next paragraph describes the technology that can be used, in order to successfully determine someone's location in an Indoor environment.

Indoor Positioning System (IPS): is the technology used to find a user's location with high accuracy in indoor spaces. An IPS is operating continuously, meaning that it receives data of someone's location in real time, unless of course the system is disabled by him. In many cases though, the absolute position of the user is not necessary and an approximate position would also cover our needs *(i.e user A is inside a building and wants to know only the room in which user B probably is)*. In that case there is no need for high accuracy in the IPS.

IPS Functionality: When using an IPS, the user's position is shown over a map *(i.e two or three dimensional)*, which is a simulation of the space where the user is. In order to make this correlation of real-time location and projection to the map, the creation of an *Indoor Map* of the space is required to be made beforehand. This procedure is called *Indoor Mapping* and is a separate research study. Moreover, when the map of an indoor space exists and a user wants to get directions of how to go to another point on the map, an *Indoor Path* planning procedure is required. Navigation of a user in an indoor space is done in terms of avoiding

obstacles like walls and finally reaching his destination in the minimum possible time. The path on the map is created using various mathematical models for calculating the shortest path. Therefore, creating an IPS may consider the sub problems of (*1) Indoor mapping (2) Indoor Positioning and (3) Indoor Path Planning.*

4.2 State of the Art

In sequence, *Section 4.2* initially presents three(3) of the most known techniques that are used today in an IPS for discovering someone's position in an Indoor environment (*i.e Subsection 4.2.1)*. Moreover, in *Subsection 4.2.2* some of the most common Location Positioning Technologies are also analysed.

4.2.1 IPS Positioning Techniques

Trilateration: In simple terms, Trilateration is a mathematical technique in which the location of a point in space is calculated using the distances from such a point, to a series of known geometrical entities *(i.e a sphere or a circle).* Thus, in our case the requested location can be calculated from 3 or more signals that are transmitted from different antennas. Trilateration applies only if the location of the antennas emitting the signals is known and then the target location can be calculated in relation to their location by taking into account the Signal's Strength. In other words, the Signal Strength is converted to Distance from the signal transmitter. However, this technique is known to be obstacle sensitive, meaning that it can be affected by obstacles that may exist in the examined Indoor Space, thereafter it is mostly used in Outdoor Positioning Systems *(i.e open space)*. Below in *Figure ?*, a paradigm of the aforementioned technique is illustrated.

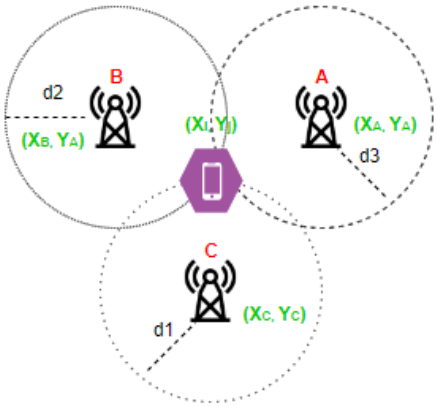


Figure 16. Trilateration

Triangulation: By definition, in trigonometry and geometry, triangulation is the process of determining the Location of a point by forming triangles to the point from known points. In our examined case, it helps us to determine a user's location based on the angle of our reference points *(i.e the signal transmitters)*. As in the previous technique, same here, the exact position of the reference points must be known in order to locate the exact position of a device in the Indoor Space. This Positioning technique works if we have two synchronised transmitters and the receiver can measure the time difference between the arrivals of the signals *(i.e TDoA)*. The receiver does not need to be synchronised with the transmitters. This technique is also used by GPS to calculate the target location. Synchronisation in the case of GPS is achieved using individual clocks on the transmitting satellites. Moreover, with the advent of MIMO *(i.e Multiple Input Multiple Output)* technology, in which multiple antennas are used on the devices, this technique has received particular interest, as it is possible to calculate the location of various receivers with just a single transmitter with multiple antennas. Below in *Figure ?*, a paradigm of the aforementioned technique is illustrated.

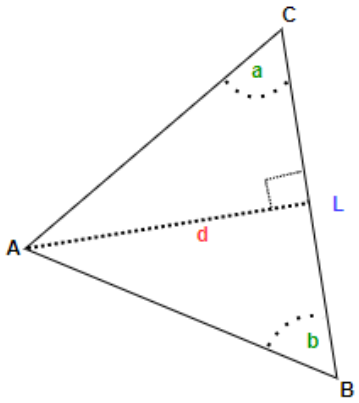


Figure 17. Triangulation

** *Last but not least, the calculation of Distance 'd' is:* $d = \frac{L\,sin(a)sin(b)}{sin(a+b)}$

Fingerprinting: This is considered to be the most popular approach among the ones described earlier in this Section. The technique consists of two(2) main phases, which are described next: *(1) Online Phase:* With fingerprinting a space is mapped by taking measurements from several points (i.e Points of Interest) and then a database could be created storing those. Then, the calculation of a target location in space is achieved in real time by comparing the measured location with the stored measurements. So far, the Fingerprinting technique has proved to be more accurate than the other techniques and this is why it has become very popular for Indoor Positioning System implementations. A technology's drawback which is important to be noted here, is that it is necessary to initially map in advance the required space that is going to be used and afterwards to store all the measurements in the database. This requires continuously updating the database in cases where 1 or more changes are made, for example if a WiFi Access Point has moved to another location or its SSID name has changed. Today there are various algorithms used, in order to more effectively match the measurements during the Online Phase. To this extent, probabilistic algorithms describe the signal strength with a probabilistic distribution function and use a Bayesian system to estimate the user's location. The most common techniques of deterministic algorithms are based on data mining and machine learning algorithms with the most famous being the K-Nearest-neighbor algorithm also known as the *KNN Algorithm*. *(2) Offline Phase*: during this part, the localisation application scenario is that the user holds a mobile device, measures RSS reading of some APs which can be sensed, and the user's location is shown on the map or illustrated as a message. The user's location estimation is done by comparing the similarity between the RSS values measured at the mobile and the fingerprint map. In this paper, the compressive sensing theory is used for reconstructing the fingerprint map. In the process of fingerprint matching, the RSS value is used as the new fingerprint to compare the measured value to the database, to find the most matching fingerprint, and to realise the localisation.

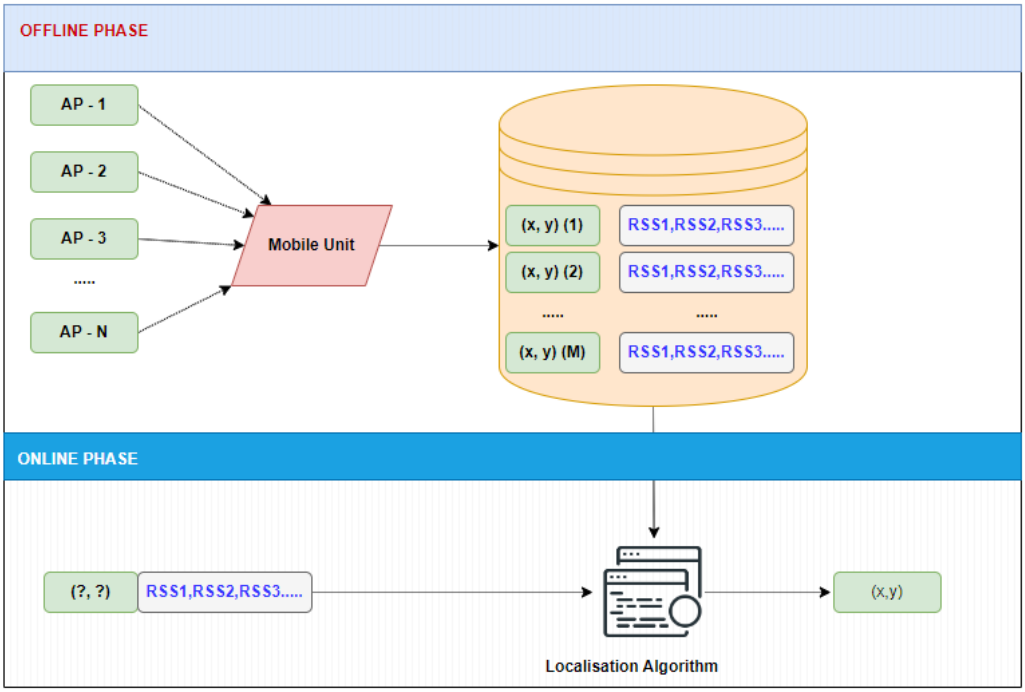


Figure 18. Fingerprinting

4.2.2 IPS Technologies

The problem of finding the location of a mobile device in an Indoor Space has been approached using various technologies with the aim of replacing the GPS that cannot be used with high accuracy. Some of these technologies are wireless *(i.e Radio Frequency)* technologies *(i.e WiFi, Bluetooth, Ultra-Wide band)*, Inertial technologies, Audible and non-Audible Sound technologies, Light and Vision-Based technologies. Among them, the WiFi technology is the most commonly used technology for Indoor Positioning Systems. Its main advantage is its universality, meaning that someone does not need to install new machines or infrastructure to use it. To this extent, next we present some of the most common Location Positioning Technologies, along with their metrics and measurements that could be extracted from utilising each one of them for different positioning case scenarios. *Wifi (Part of our approach):* The most common approach using this technology is the Fingerprinting technique, which creates an a priori map of points with their Signal Strength measurements, and then in real time, the signal received by the user is compared with the stored measurements, thus making it possible to deduce his actual position in the specified area. Moreover, Time of Arrival *(i.e ToA)* of the signal can be used to find the location of a user. In ToA, the transmitter includes the time when the signal is transmitted in the radio packet, and the receiver calculates the time spent on the transmission. With this information, the receiver can calculate the distance *(i.e d)* between two Access Points, with the prerequisite that the nodes are synchronised. Moreover, it was observed that the existence of APs with stronger Signal Strength is more important than the number of Access Points as the weakened signal shows significant fluctuations in its values, as the use of such signals affects the accuracy of the measurements. The main disadvantage of using the WiFi technology refers to the signal propagation model *(i.e due to the existence of obstacles around the area)* and the existing infrastructure *(i.e number and location of APs)*. *Bluetooth:* Bluetooth has become a competitor to WiFi technology in terms of Indoor Localisation due to its upgrade with the BLE sensors. It is used mostly to exchange data over short distances using UHF radio waves *(i.e frequency bands 2.4-2.485 GHz)*. This technology makes it ideal for usage in closed spaces (*i.e Indoor Areas)* where someone can easily place them around the area. Due to this, there are several commercial applications that use this particular technology to achieve Indoor Localisation. In such Indoor Positioning

Systems, the model uses the Received Power from the Bluetooth Low Energy sensors, to finally determine the user's location in the area based on that.
Visual Based (Part of our approach): By using the camera of a mobile device, the surrounding area can be captured and then find a target location, by matching these images with the current image of a camera. Also, by using specific points(i.e Access Points) as landmarks, a target location can be calculated in relation to the distance from these points. Another visual technology that has received a lot of attention in recent years is the Augmented Reality (i.e AR). AR, is an interactive experience that combines real world and computer-generated content. AR can be defined as a system that incorporates three basic features: *(1)* a combination of real and virtual worlds *(2)* real-time interaction and *(3)* accurate 3D registration of virtual and real objects. Augmented reality has been used a lot in recent years in navigation applications. For better accuracy of the location of "augmented" objects in space, special emphasis was placed on algorithms that utilise vision sensors, i.e. the camera of mobile devices, to locate and store various points of interest in space and map the environment. This helps to compare and recognise points in space and not whole images with the camera. The main algorithm used in AR systems is *Tracking and Registration* which achieves the recognition and registration of an environment based on discrete point recognition. To this extent, an AR tracker is a specific pattern or image that an AR app can recognise. Once the app finds the pattern, it constantly 'tracks' the position of the pattern in real world space, so the application can finally accurately place a virtual object onto the tracker. Another approach that uses the camera of the user's mobile device, is by scanning QR codes. First, it is required to create a map of an indoor space or a building and use some spots as points of interest, where a QR code is assigned to each point. Then navigation is achieved by scanning these QR codes that represent the current position of the user over the map. So, with the knowledge of the current position, navigation to other destinations can be achieved by calculating the shortest route to a target point and directions are given based on the mobile's inertial sensors.
Infrared: Infrared *(i.e IR)* communications use nonvisible light to transmit data. Unlike other wireless communications based on Radio Frequencies, this technology has a great disadvantage by not being able to penetrate through obstacles. Therefore, this technology requires Line-Of-Sight (i.e LOS) communication for accurate results.
Geomagnetism: Magnetic maps have been observed to maintain stable values over a long range of time. For this reason they have been studied in situations where other wireless technologies are not available (i.e underground implementations).
Inertial Sensors System: Navigation is achieved by giving an initial position to the inertial system *(i.e for example the last GPS measurement before entering an indoor space)* and calculating a target position from the difference of that position and the position measured by the sensors. However, the measurements from these sensors suffer from the drift problem. With the continuous use of these sensors, the small differences observed add up and we have a deviation *(i.e drift)* from the real data that increases exponentially. Therefore, if we rely only on these sensors then there will be large deviations on long distances. Measurements on mobile devices are also affected by magnetic disturbances and irregular movements of the user's body and hands. Therefore, although these sensors give accurate measurements, it is preferred to be combined with other technologies that periodically update the user's location and nullifies the drift that occurs. For example, they are used together with GPS in many aircrafts. Estimating a future position given an initial position associated with speed and direction is one of the oldest methods of navigation, generally called "dead reckoning". In the context of an inertial sensors system, Pedestrian Dead Reckoning *(i.e PDR)* uses only inertial sensing *(i.e accelerometers, gyroscopes, sometimes magnetometers)* to estimate speed, steps, stride length and direction, which then are converted into distance. This means that no external information is needed except the information taken by the sensors from the mobile device.
Mobile Radio technology (Part of our approach): Mobile radio signals cover a larger area compared to WLANs but with lower accuracy. To achieve Indoor Localisation with mobile radio signals, there are plenty of approaches that can be used like, measuring the Received Signal Strength , the Angle of Arrival, the Time of Arrival or the Time Difference of Arrival. Moreover, the Fingerprinting technique can also be applied here, since the user's location can be calculated by using the Triangulation *(i.e this can be done using various*

techniques such as through radial distance, direction or receiving a signal from two to three different points and then assessing the exact location by overlapping of the three radial distances) technique of the Downlink and Uplink signals of a mobile device.
Ultra Wide Band: This technology uses very low energy to transmit data over short distances, at a high data rate and over a wide spectrum. For Indoor positioning procedures, it uses the time difference of arrival *(i.e TDoA)* signals to calculate the distance between a reference point and a target. Another implemented system is the Ubisense *(i.e tracks the precise location, movement and interaction of things within programmable spaces)*, where the user carries a tag that transmits a UWB signal to fixed sensors, that can measure the user's location using the time-of-arrival *(i.e ToA)* method.
Audible Sound: Using Audible Sounds is another approach to Locate a user in a given area. Sound signals, consisting of pressure waves that propagate in the air at a much slower speed than electromagnetic signals. The time difference between the emission and arrival of a signal is used to measure the exact location.
UltraSonic Sound: Ultrasonic Positioning Systems use building materials and air as propagation medium. The relative distance between different devices can be calculated by measuring the arrival time of the ultrasonic pulses, traveling from the transmitters to the receivers.

4.2.3 Related Works

As the Indoor Positioning Systems attracted a lot of attention throughout the latest years, new research and development topics have emerged around the globe. So far different kinds of approaches and techniques have applied, yet new milestones arise as the accuracy of systems like this is becoming a top priority. This makes a lot of sense, since the reliability of such systems depend mostly on how accurate they could be, in terms of finding the exact location of a user in an Indoor Environment. Consequently, the subsection here presents in detail some of the works done so far on the topic and finally a brief description of our approach, on the matter, as well.
Many technologies like the ones described in the previous subsection are used today in Indoor Positioning System implementations. Compared with those, the Traditional Radio Frequency Identification *(i.e RFID)* technique has also been studied. The RFID indoor positioning algorithm *LANDMARC* utilises a Received Signal Strength *(i.e RSS)* indicator to track objects. However, the RSS value can be easily affected by environmental noise and other interference. In [31], an indoor positioning method using passive UHF RFID is presented. Moreover, a novel Indoor Positioning Algorithm based on the Bayesian probability and the K-Nearest Neighbour *(i.e BKNN)* is presented. A Gaussian filter was used to filter the abnormal RSS values and Bayesian estimation in combination with the KNN algorithm improved the positioning system's accuracy. In conclusion, the average location error of the algorithm presented in this work was about 15cm. The work presented in [32], focuses on the analysis of several typical fingerprint localisation algorithms including the NN, KNN, WKNN. The article presents an optimisation algorithm of KNN and introduces an Indoor WiFi signal propagation model. Theoretical analysis and experiment results showed that the KNN optimization algorithm has a certain degree of enforceability and the original algorithm has improved greatly in terms of positioning accuracy. To this extent, in order to improve the precision of the Location Fingerprint technique and optimize the performance of such systems, more proposals of improved weighted KNN algorithms have been applied. In [33], is presented an improvement of the WKNN Location Fingerprint Algorithm. The proposed algorithm is divided into 2 stages, the offline and online phase. In the former one, the Indoor Positioning Area is divided into several GRIDS and each one is sampled repeatedly. The steps followed are: area division, fingerprint data acquisition, data filtering and finally construction of the Fingerprint database. On the other hand, in the offline phase the client obtains the fingerprint data, uses fingerprint to match with the database and finally by applying the improved WKNN algorithm the 'k' GRID Points closest to the fingerprint data are obtained and it calculates the user's physical location. As a conclusion, the work utilises the variance of Access Points *(i.e AP)* signal strength to define the stability of

each AP and calculates the weight of WKNN algorithm based on the stability of each AP and the correlation of each divided GRID. The simulation results showed that the proposed algorithm has better performance than the W-KNN algorithms, which indicates that the weight of the composition in the W-KNN algorithm affects the positioning precision . Wi-Fi indoor positioning depends on the Wi-Fi signal to get indoor location information, which is of great use and significance to the indoor positioning application. Fingerprinting based on Wi-Fi Received Signal Strength Indicator *(i.e RSSI)* has been widely studied in recent years for indoor localisation. While current algorithms related to RSSI Fingerprinting show a much lower accuracy than multilateration based on time of arrival or the angle of arrival techniques, they highly depend on the number of Access Points *(i.e APs)* and fingerprinting training phase. In [34], is presented an integrated method by combining the Deep Neural Network *(i.e DNN)* with improved K- Nearest Neighbour *(i.e KNN)* algorithm for indoor location fingerprinting. The improved KNN is realised by boosting the weights on K-nearest neighbours according to the number of matching APs. This overcomes the limitation of the original KNN algorithm on ignoring the influence of the neigh-boring points, which directly affect the localisation accuracy. The DNN algorithm is first used to classify the Wi-Fi RSSI Fingerprinting dataset. Then these possible locations in a certain class are also classified by the improved KNN algorithm to determine the final position. The proposed method was validated inside a room within about $13*9\ m2$. To examine its performance, the presented method has been compared with some classical algorithms, i.e., the Random Forest *(i.e RF)* based algorithm, the KNN based algorithm, the Support Vector Machine *(i.e SVM)* based algorithm, the Decision Tree *(i.e DT)* based algorithm, etc. The real-world experiment results indicated that the proposed method is less dependent on the density of access points and Indoor Radio Propagation interference. In other works hybrid models of the basic KNN algorithm have also been applied in order to achieve high degree of position accuracy. Next, is described such a case where a combination of the Fuzzy C-Means Clustering algorithm *(i.e FCM)* and the KNN resulted in increasing the system's locationing accuracy. Generally, the FCM algorithm works by assigning a membership to each Data Point corresponding to each Cluster center on the basis of distance between the cluster center and the data point. The more data is near to the cluster center the more is its membership towards the particular Cluster center. In [35], by applying the proposed algorithm through KNN, 'k' Points are firstly chosen as the data samples of FCM based on the Received Signal Strength *(i.e RSS)*. Then, the 'k' Points are classified into different Clusters through FCM based on RSS and the position coordinates. According to the rules proposed in this work, some RPs are reselected for indoor location in order to improve the location precision. Simulation results indicated that the proposed KNN-FCM hybrid algorithm outperforms KNN when the location error is less than 2 meters. Another issue related with the IP systems, is the variations of the Received Signal Strength Indicators *(i.e RSSI)* in wireless networks, especially when it comes to locate the positioning of moving objects indoors. Those variations could be present due to the absence of line of sight *(i.e LOS)* as a result of the lack of directional antennas, and to the propagation phenomena of electromagnetic waves, such as path loss, shadowing and multi-paths. Thus, the theoretical techniques used in some works without the addition of the RSSI variation cannot express the loss of signal power in practical tests, as they are limited to simulated experiments. To this extent, [36] investigates and proposes a method of indoor localisation, in order to improve the accuracy of positioning that is compromised by those variations. The approach presented in this work utilises a Quartile Analysis *(i.e QA)* for the data pre-processing in combination with the KNN classifier. Quartile analysis is a statistical method used to assess the central trend and data dispersion. Quartiles partition a partially ordered set *(i.e poset)* into four equal parts. The first quartile (Q1/4) or lower quartile, delimits the 25% of the lowest observations; the second quartile (Q2/4), or median, separates the 50% smallest from the 50% largest observations; and the third quartile (Q3/4), or upper quartile, delimits the 25% largest observations. The results of this work, showed that the possibility of accurately locating in indoor environments with problems of variation of the RSSIs in wireless networks is verified.

In [37], another Hybrid Model based on the Constraint Online Sequential Extreme Learning Machine *(i.e COSELM)* Classifier with Adaptive Weighted Sparse Representation Classification *(i.e WSRC)* and KNN is

proposed, for a WiFi-based IPS. The Model exploits the speed advantage of COSELM to reduce the computational cost, and the accuracy advantage of WSRC to enhance the classification performance, by utilising KNN as the adaptive sub-dictionary selection strategy. The estimation of the position takes in account and position *(i.e longitude and latitude)* in a hierarchical and sequential approach according to the discriminative criterion to the COSELM output. If the classifier result is unreliable, AFARLS uses KNN to achieve the best relevant sub-dictionary. Finally, The sub-dictionary is fed to WSRC to re-estimate the Indoor Building and Floor, while the position is predicted by the Extreme Learning Machine Regressor. The results of this work concluded that the proposed Model can offer real-time performance with high accuracy, by exploiting the advantages from the original ELM, SRC and KNN algorithms in order to tackle their respective drawbacks which render their practical applications in Indoor Positioning System implementations.

As a final example of the different kinds of approaches that could be followed in order to increase an IP System's location accuracy, is described next. So far it has been proven that the application of only the KNN or the WKNN algorithms are not efficient enough to achieve high levels of accuracy in such systems. Moreover, in indoor areas with only a few WiFi APs the deviation of someone's exact positioning in space could be very high. Based on the former, *[38]* describes a Hybrid Indoor Positioning Algorithm for Cellular and Wi-Fi Networks. As in most cases, also here the proposed model consists of two(2) parts, the Offline and the Online phase. During the first phase, a Database is created storing the Fingerprints by using principal component analysis *(i.e PCA)* and interpolation methods. Next, in the Online phase for the positioning, it is used the Back Propagation Neural Network Positioning Algorithm *(i.e BP)* which has been optimised by the Adaptive Genetic Algorithm *(i.e AGA-BP)*. The algorithm uses Cellular Network Positioning to divide sub-regions and then uses Wi-Fi networks to further improve the level of accuracy. In conclusion, the results of these work experiments showed that the positioning error of the AGA-BP positioning algorithm in the whole area is 1.42m, which is improved by 32.7% and 28.3%, compared to WKNN and BP network, respectively. The average positioning error of Hybrid Indoor Positioning *(i.e WiFi + Cellular)* scheme is 1.70m, which is 55.96% lower than using Wi-Fi network only.

The Section following next, presents in detail our approach on the matter. Briefly, our IPS implementation utilises a Hybrid model of WiFi, Cellular and Visual Based Technologies *(i.e Augmented Reality and Cloud Anchors)* for the Mapping and Positioning techniques.

4.3 System Model

The following Section presents an analytical description of our System Model's service. The proposed solution is an Indoor Positioning based model, that utilises the fingerprinting technique, in order to map a space in terms of the available WiFi and Cellular signals around the user's position. This can be achieved at specific location points in the area space we choose to map. For this purpose we use a Location Server *(i.e MEC Server*) which is responsible for the Indoor Location Management service. To this extent, next we describe how this service works. The MEC Server is responsible for gathering and afterwards storing on a database, all the important data coming from the Agents side. These information includes a unique identification number *(i.e ID)* of the Agent, a Timestamp referring to the exact time of the acquired data, the coordinates *(i.e X,Y,Z)* in the area, based on a reference point (0,0,0) which is specified from the system and a Set of measurements which is of dynamic size *(i.e. not static, can have WiFi or Cellular signal measurements etc.)*. Each Agent, considering the fact that he has Objective Information about where he is *(i.e position in area)*, gives this data measurements to the Location MEC Server. The Location Server then accumulates this information, sends it to the database which is responsible for storing the data acquired, arranges them and finally assigns their weights based on the data Timestamp and the user's reputation in the system. On the other hand, while using the Service in real time, Client(s) can also send their data to the Location MEC Server. The reason behind this action is that the Server is also responsible for initiating a

Matching process, where based on the data information gathered *(i.e Radio, Visual mapping)* from both sides *(i.e Agent/Client)* he can define the user's exact position in space by applying Interpolation and Extrapolation mapping techniques. These techniques will be described later in Section *(i.e Section IV)*. Moreover, below is described the proposed System's Service in steps, along with a Figure *(i.e Figure 19)* presenting its Architecture.

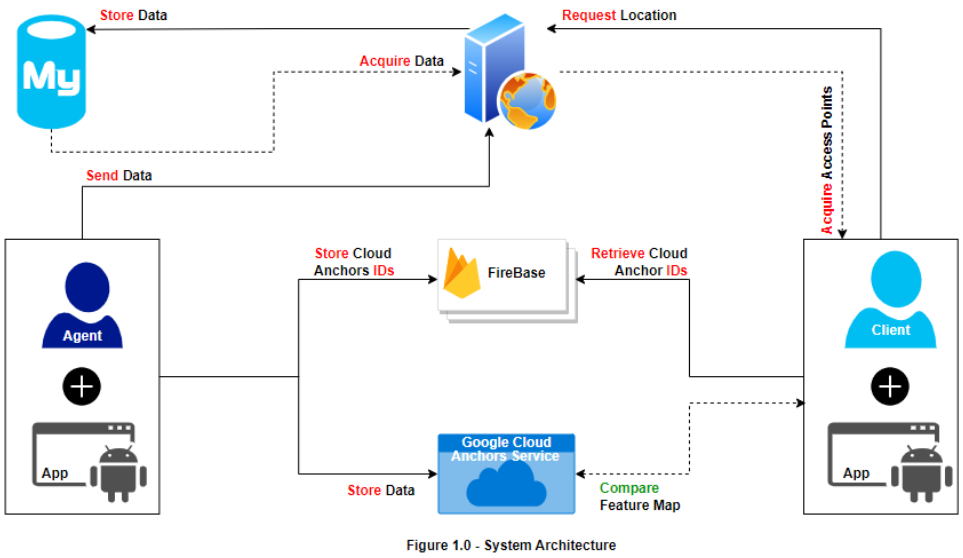


Figure 19. System Model

4.3.1 System and Service Description

Roles

Before moving on describing the steps of the process presented in *Figure 1.0*, it is necessary to clarify the purpose of each *Role* in the System. An *Agent* is a user who acts as a Mapper *(i.e fingerprinting technique)*, meaning that he is responsible for defining and providing to the Server all the important information needed for the Matching process. On the other hand a *Client (i.e Locator)* is a user who locates his exact position in the defined area, based on the information that an Agent already created and provided to the *Server*. Moreover, the implementation of our proposed system consists of three(3) basic components: *Component 1)* In order to Map the reference points and to later search for them once an interior space has been fully mapped, a custom mobile application is used. The application was developed just for Android software at time being and it is only compatible with devices that support Augmented Reality *(i.e AR - library of ARCore)*. *Component 2)* System's second component is the *Server* which is responsible for gathering and storing the data extracted from the reference points. The Server acts as an HTTP Server connected to a Local Database and serves requests in order to determine the location of a user *(i.e Client)*. *Component 3)* The third building block of our system, is an online Database that stores the unique identifiers *(i.e IDs)* taken from each reference point. To this extent, Google provides a very easy way to create a

database and to connect it with our custom application, the Firebase (i.e API that helps us sync and store data in real time).

Process Steps

Step 1: In the first step, the Agent initiates the Android application on his device and he can start the Mapping process. As mentioned already the proposed system uses the Fingerprint technique to map an area in terms of WiFi and Cellular antenna signals available to the user. The mapping procedure allows the Agent to place an Augmented Reality Object on space, marking its location. The Agent applies this technique to specific points of interest he chooses to map in the area. The data that is stored in each point is a unique ID and signal strength measurements taken from each WiFi access point and cellular cell around the user. At the same time while recording the signals, he can also place a Cloud Anchor at the exact point being mapped. This allows feature points to be detected for a specific amount of time *(i.e 30 seconds*) from the Android device mobile camera. The mapping process is completed when the signal at each predefined point is measured and when all cloud anchors are placed on the map (i.e indoor area). It is necessary to note that for each Cloud Anchor a unique identifier is generated as reference within Google's databases and is large in size. For this reason, we store a short form of this identifier *(i.e ID)* in our local database and keep the full form along with the corresponding short form in our Firebase database. To this extent, the full form identifiers of the cloud anchors along with their corresponding short form are both stored in a database created in Firebase. When it is required to find the location of a user, our application calls the cloud anchors service to compare the image from the device camera with the feature points of the cloud anchors we request. The IDs of these cloud anchors are then pulled directly from our Firebase database.
Step 2: In this step, all the important information about the signal measurements is now stored in a database located on our Location Server. Moreover, the unique IDs from the placed Cloud Anchors are also stored in another one called Firebase.
Step 3: Next, a user could now act as Client, again through our custom Android application. By using the data information that was created and stored by the Agent(s) in the previous steps, he can now find his position in an indoor space after this has been already properly mapped. This works in two distinct phases: *1)* In the first phase the device uses information about available mobile radio signals and available WiFi networks to locate the closest reference points to the user. These reference points are retrieved by the server and the best 20 are returned sorted to the user. Then the device starts and identifies the image from the device camera with these points through the Cloud Anchors service. *2)* The second phase uses the device camera and augmented reality capabilities provided by ARCore. After matching at least one point, the application calculates the user's location based on the distance from that point and the signals available. Therefore, the application considers the user's location to be the point that has the best match for the user's data. If no cloud anchor is detected then the user's location is derived only from WiFi and mobile radio signals.
Step 4: The Mapping and Locating procedures are both completed and the Client is now fully informed about his exact position in the predefined area used for this purpose.
The following Subsection, presents a more clear analysis with respect to the Mapping and Locating procedures, that both explained previously in this Section.

4.3.2 Signaling process (Service flow)

The system specifications are described below in detail for each component. The application has two separate functionalities. The first one is the mapping function *(i.e the fingerprinting technique is referred to as the offline phase - Agent/Server side)* and the second one is about locating the user *(i.e online phase Client/Serverside)*.
Server: On the Server side we have a Database keeping all the Access Points that have been created and used in the mapping procedure. The database is of MySQL type and contains a table named "readings". In this database Table, the room, the coordinates, the number of the Access Points and finally the values of

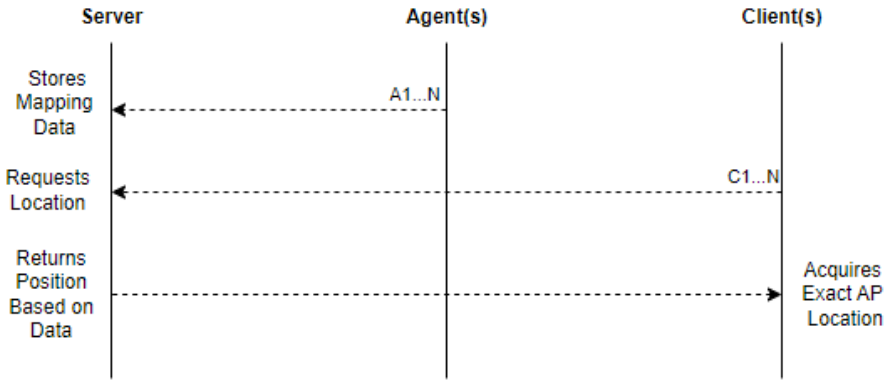


Figure 20. Service Flow

each measured signal are all stored. Those measurements and metrics will be described in detail next in Subsection 3.3. The server has two Java Servlets *(i.e A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.)* to serve the user's requests about storing new reference points *(i.e APs)* or for restoring information about the exact location. The first Servlet responds to requests about adding a new measurement *(i.e inserting a new reference point into the database)*, while the second implements the data matching algorithm and returns the reference points *(i.e the best 20)* that are closer to the user along with their coordinates.

Agent: When a user wants to participate in the Mapping procedure, initially he has to use our Android Indoor Positioning Application. Once a specific button in our application is selected, the camera of the user's device is automatically activated. A prerequisite is that the device has ARCore installed. When the camera is activated, the user has at his disposal another button that allows him to place an "Anchor" at the location of his mobile device in an Augmented Reality environment. Finally by pressing the button specified for placing the Anchor in the area, a notification box appears on the screen, asking the user to fill in with information about his referenced Access Point that is going to be placed. These information will also be described in detail later, in Subsection 3.3. After the required data has been entered by the Agent, the process of recording the feature points around the placed anchor begins. This process takes a specific amount of time *(i.e 30 seconds)* and there is a timer on the screen that shows the remaining time for the user to record the space around the anchor. Ideally, within 30 seconds the device should be rotated slowly and steadily so that the image is as clear as possible. Finally, after 30 seconds, the Agent can upload the feature points *(i.e APs)* to the Google Server in order to be used later for the Locating procedure by the Client.

Client: With the same way a user can identify as a Mapper, by using our Custom Android Application he can also play the role of a *Locator (i.e Client)*. The application activates the camera of the device and transports the user to an Augmented Reality environment. With the press of a specified button, he initiates the process of locating a user in space. The process consists of two parts: 1) The Wi-Fi and Mobile Radio signals available to the user are measured and then sent to the central Server. The server, using another Servlet, compares the measurements with the reference points stored in the database. Up to 20 points with their coordinates are returned back to the application. The IDs of these points are stored in short form, so Firebase retrieves their full form IDs. The application uses the IDs of these points to match the camera

image with the stored points. This process takes place on Google servers automatically. If there is a successful match, then the Anchor that was placed during the mapping procedure is displayed and an Augmented Object is created showing all the details about the specific Access Point *(i.e X, Y coordinates and distance 'd' from the user's device)*. At this point, the Mapping and Location Procedures between the Agent/Server/Client has successfully been completed. The Signaling Flow Diagram *above (i.e Figure 13)* presents an abstract illustration of the communication links between the parties involved in our Indoor Positioning System.

4.3.3 Metrics & Measurements

The Subsection here, summarizes the measurement capabilities of some of the Indoor Positioning System Technologies, that have been described in *Section 2*. The paragraphs following next, presents an extended analysis of each technology's general specifications and metrics.

IEEE 802.11: is a set of standards supporting wireless local area network (i.e WLAN) access in the 2.4, 3.6 and 5 GHz frequency bands. The baseline version of the standard, i.e. IEEE 802.11-2007, has been recently updated to its current version (i.e. IEEE 802.11ax), aiming to include a series of amendments developed over the past few years. Emerging in 2021, Wi-Fi 6 works more efficiently in crowded networks. Earlier Wi-Fi devices can detect and bypass Wi-Fi 6 packets. Wi-Fi 6 supports 1024QAM modulation *(i.e four times Wi-Fi 5's 256QAM)* as well as two multi-user uplink and downlink transmission methods *(i.e MU-MIMO and OFDMA)*. Wi-Fi 6's theoretical maximum speed is 9.6 Gbps compared to 3.5 Gbps for Wi-Fi 5 (i.e 802.11ac).

Measurements

Received channel power indicator (RCPI) : An indication of the total channel power (signal, noise, and interference) of a received IEEE 802.11 frame, transmitted from STA s to STA s', measured on the channel and at the antenna connector used to receive the frame at STA s'.
Average noise power indicator (ANPI): A MAC indication of the average noise plus interference power measured by STA s, when the channel is idle as defined by three simultaneous conditions: (1) the Virtual CS mechanism indicates idle channel, (2) the STA s is not transmitting a frame, and (3) the STA s is not receiving a frame.
Received signal to noise indicator (RSNI) : An indication of the signal to noise plus interference ratio of a received IEEE 802.11 frame, transmitted from STA s to STA s'. RSNI is defined by the ratio of the received signal power (RCPI-ANPI) to the noise plus interference power (ANPI) as measured on the channel and at the antenna connector used to receive the frame.
Max transmit power (MTP): The Max Transmit Power field is a 2's complement signed integer and is 1 octet in length, providing an upper limit, in units of dBm, on the transmit power as measured at the output of the antenna connector to be used by STA s on the current channel. The maximum tolerance for the value reported in Max Transmit Power field shall be 5 dB. The value of the Max Transmit Power field shall be less than or equal to the Max Regulatory Power value for the current channel.
Transmit power used (TPU): The Transmit Power Used field is a 2's complement signed integer and is 1 octet in length. It shall be less than or equal to the Max Transmit Power and indicates the actual power used as measured at the output of the antenna connector, in units of dBm, by STA s when transmitting the frame containing the Transmit Power Used field. The Transmit Power Used value is determined any time prior to sending the frame in which it is contained and has a tolerance of ± 5 dB.
WLAN Metrics (Localisation): With respect to our Android Application, the AndroidOS allows developers to create applications that use the mobile's network card. Information about the nearby WiFi networks can be obtained without requiring connection of the device to the target WiFi. This is achieved by receiving the periodically broadcasted packets by the WiFi access points (APs), or by broadcasting a request message to

all nearby WiFi APs to provide information. The information obtained by such packets is the **BSSID** (Basic Service Set Identifier), the **SSID** (Service Set Identifier), the **RSSI** (Received Signal Strength Identifier), the **FREQUENCY** it transmits and the **IP ADDRESS** of the router. An application can monitor these packets and use this information to measure the signals of each router in the area for a long period of time. These measurements would be very useful in terms of an IPS system as the user moving in an indoor space could know his position relative to the routers.

LTE/4G: LTE stands for Long-Term Evolution and is a registered trademark owned by ETSI (European Telecommunications Standards Institute) for the wireless data communications technology and a development of the GSM/UMTS standards. However, other nations and companies do play an active role in the LTE project. The goal of LTE was to increase the capacity and speed of wireless data networks using new DSP (i.e digital signal processing) techniques and modulations that were developed around the turn of the millennium.

Measurements

Reference signal received power (RSRP): The linear average over the power contributions (in [W]) of the resource elements that carry cell-specific reference signals within the considered measurement frequency bandwidth. For RSRP determination the cell-specific reference signals R0 shall be used while the UE may use R1 in addition to R0 if it is reliably detected. The reference point for the RSRP shall be the antenna connector of the UE.

E-UTRA carrier received signal strength Indicator (RSSI): The linear average of the total received power (in [W]) observed only in OFDM symbols containing reference symbols for antenna port 0, over Rc, DL number of RBs by the UE from all sources, including co-channel serving and non-serving cells, adjacent channel interference, thermal noise etc. RSSI is not reported as a stand-alone measurement rather it is utilised for deriving RSRQ.

Reference signal received quality (RSRQ): The ratio Rc, DL×RSRP/(E-UTRA carrier RSSI) where Rc, DL is the number of RB's of the E-UTRA carrier RSSI measurement bandwidth. The measurements in the numerator and denominator shall be made over the same set of RBs. The reference point for the RSRQ shall be the antenna connector of the UE.

Downlink reference signal transmitted power (DL RS Tx): The linear average over the power contributions (in [W]) of the resource elements that carry cell-specific reference signals which are transmitted by a tagged cell within its operating system bandwidth. For DL RS TX power determination the cell-specific reference signals R0 and if available R1 can be used. The reference point for the DL RS TX power measurement shall be the TX antenna connector.

Received interference power (RIP): The uplink received interference power, including thermal noise, within one physical RB's bandwidth of NRB resource elements. The reported value shall contain a set of Received Interference Powers for all the uplink physical RBs. The reference point for the measurement shall be the RX antenna connector.

Cellular Metrics (Localisation): Another method that was studied in our work, was by using the mobile Radio Signals and more specifically to observe the Received Signals on the mobile phone and analyze the available data of the base stations (BS). Information about all antennas (BS) within range of the user (for the specific provider they are subscribed to) can be obtained. This information includes the **Cell Id**, Mobile Country Codes **(MCC)**, Mobile Network Codes **(MNC)**, Location Area Code **(LAC)**, **connection type** (i.e between GSM, CDMA, WCDMA and LTE) and **Signal Strength**. Also, the locations of the base stations are available online on specific websites, which can also be exploited to retrieve more information. An application that collects this information (i.e Cell Id, MCC, MNC and LAC) from each antenna that is within range of the user was created and then the geographic coordinates of the BS were retrieved. Moreover, by using the Geographic Coordinates of the user from GPS, we created augmented reality tags (i.e AR **Tags**) showing the location of the BS and its distance from the user.

Bluetooth: Its implementation cost is very low, thus making it extremely adopted on the domain. Moreover, it uses extremely low power consumption and it is available in almost all of the mobile devices existing today on the market. As *measurements* for Location Calculation, it takes in consideration the

Proximity of the device, the **RSSI** metric and the **BLE Beacons**. Some disadvantages by using BLE as a Localisation Technology, is that It requires quite a few cells to achieve good accuracy (i.e thus increasing the cost). Furthermore, its coverage area reaches up to 30m and its accuracy levels fluctuate between 1 and 3 meters. Finally, signal interference is possible from other devices in the 2.4Ghz spectrum. **Visual Based:** It doesn't require additional equipment except the camera and the sensors of the device, if AR is used. A disadvantage is that it is considered to be a complex approach, since it requires a Database Server and its energy consumption is high. Its coverage area reaches up to 10m and its accuracy levels fluctuate between 0.01 and 0.1 meters. As *measurements* for **Location Calculation**, it uses **Augmented Reality** and **Image/Pattern** Recognition (i.e QR codes) techniques. **Infrared:** Even though this technology is Ideal for sensitive communications, because it does not penetrate walls, yet it is limited for use in small spaces. Communication is blocked by any obstacle between the receiver and the transmitter, thus it requires Line Of Sight. As *measurements* for Location Calculation, it takes into consideration its Received Signal Strength (i.e **RSS**). Its coverage area can reach up to 5m and its accuracy levels fluctuate between 0.001 and 0.01 meters. **Geomagnetism:** It uses stable magnetic imprints for long periods of time and It is affected by ferromagnetic objects such as doors, elevators etc. Its coverage area is considered as infinite and its accuracy levels fluctuate between 2 and 6 meters. Lastly, as *measurements* for the Location Calculation, it takes in account the **Magnetic readings** by the device's magnetometer. **Inertial Sensors:** It does not require additional equipment except the sensors of the mobile device. A disadvantage is that it calculates the relative position based on the initial position. Again as previously, its coverage area can be considered as infinite and as *measurements* for the Location Calculation it uses the **Dead Reckoning** technique *(i.e the process of calculating current position of some moving object by using a previously determined position)*. **Ultra Wide Band (UWB):** This technology meets very high accuracy, even when there are losses due to multi-path transmissions. Moreover, it can pass through obstacles and it does not interfere with other radio wave technologies. Two main disadvantages are its high cost of equipment and that metallic objects could cause Interference issues. As *measurements* for the Location Calculation, it takes on account the **ToA**, **TDOA** and **ToF methods** (i.e are methods for measuring the distance between a sensor and an object, based on the time difference between the emission of a signal and its return to the sensor, after being reflected by an object). In conclusion, the coverage area can reach up to 10m and its accuracy levels are between 0.02 and 0.5m. **Non Audible Sound (UltraSonic):** Ultrasonic technology does not interact with electromagnetic waves and does not require Line Of Sight (i.e **LoS**) communication. A big disadvantage is that it cannot penetrate solid walls and there could be false signals due to signal reflections. Interference from sounds can be mostly achieved at high Frequencies. The technology's coverage area is 10m and the accuracy levels from 0.02 to 0.5m. As *measurements* for the Location Calculation, it takes into consideration the **ToA**, **TDOA** metrics.

Below, ***Table 3*** *Summarizes the Measurements and Metrics* taken into account from each technology, when implementing an Indoor Positioning System.

IPS Technology	**Measurements & Metrics (IPS)**
WiFi	RSSI, Fingerprinting
Cellular	RSSI, Fingerprinting
BLE	Sensor Proximity, RSSI, BLE Beacons
Visual Based	Image/Pattern Recognition, AR
Infrared	RSS
Geomagnetism	Magnetometer Sensor Readings
Inertial Sensors	Dead Reckoning
Ultra Wide Band	ToA, TDoA, ToF Methods
UltraSonic	ToA, TDoA

Table 3. IPS Metrics

4.4 Algorithms

This Section includes an introduction to some of the most known Matching Algorithms used today for Indoor Positioning Systems, along with a detailed analysis with respect to their types *(i.e Subsection 4.4.1)*. Moreover, *Subsection 4.4.2* describes the Proposed Algorithms and finally Subsection 4.4.3 concludes with a discussion based on the Algorithmic Implementations that presented earlier in this Section.

4.4.1 Matching Algorithms

This document presents an Indoor Positioning System *(i.e IPS)* implementation, where its system model and roles described earlier in *Section 3*. In general, the IPS can be implemented in various ways in terms of how it calculates and locates the user's exact positioning in a defined area. The first approach in mind was to initially calculate the room in which a user is and afterwards to search for Cloud Anchors (*i.e Cloud Anchors connect real world locations with digital content that anyone can access from compatible mobile devices when using AR)* in that room. After taking some measurements, we decided to try to reduce the

number of Cloud Anchors that were placed in each room. This choice was made in order to reduce the amount of time for the Mapping procedure. Next, we present the three(3) different approaches on the matter.

Approach 1. Full Grid - Room Driven : Fingerprinting technique on the WIFi and Cellular signals is used, in order to find the room in which the Agent is and later to find the exact location of the Client in relation with the Cloud Anchors that have been defined around the area. In this approach, the location of the user is considered to be the reference point at which he is closer to. The distance between the user and each Cloud Anchor is calculated by our custom mobile application. Following, some various algorithms that help us to determine the room in which a user is, are presented and described next.
Process: For each reference point that has been successful mapped in the area, a Fingerprint is created. This Fingerprint is then stored in the Database of our Central Server in the format **{RSSSih, RPi}**. The Reference Point *(i.e **RPi**)* indicates the position of the **ith** Reference Point *(i.e the room it is located)* and **RSSih** is the vector of the **RSS** measurements of the **ith** Point, which is **{RSSi1, RSSi2…RSSin}**. When each Fingerprint is created, all the data described previously refer to the Mobile Radio and WiFi Signals that were measured during this process. Thereafter, when a user searches for his location, a new Fingerprint is created of the form **{RSSu1, RSSu2,...RSSug}**, where **RSSu** is the signal available to the user, each time he tries to find his exact position in the determined space. As already mentioned in *Section 4.2*, there are many technologies and implementations that can be used to achieve a user's positioning in space. Thus, there are many ways to achieve calculation of someone's location or the closest Reference Point to him, based on his Fingerprint. The most common way today is by exploitation of the K-Nearest Neighbours *(i.e KNN)* algorithm or different variants of it.

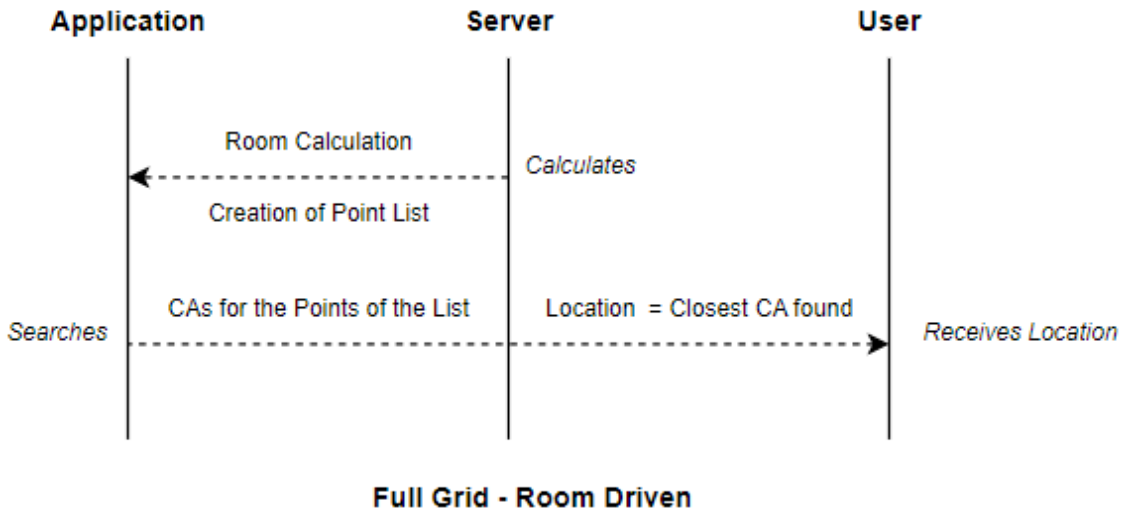


Figure 21. FG/RD Scenario

K-Nearest Neighbours Algorithm (KNN)

The k-nearest neighbours algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another. The Algorithm selects the Fingerprint with the minimum Distance *(i.e **D**)* from the user based on the Received Signal Strength *(i.e **RSS**)*. The most common metric for calculating the closest distance is the Euclidean distance between metric vectors. To this extent, the K Nearest Neighbours are selected by measuring the Euclidean distance from each Reference point. In this case if a signal is not available or absent, the value of **RSSih** is considered to be "0". After calculating the distance from all Reference Points *(i.e **Di**)*, the results are sorted and the "K" shortest distances are selected. The efficiency of the Algorithm depends on the "K" value. The most common value is the K=n, where "n" is the number of Reference Points. Last but not least, the algorithm presented here, although widely used, comes with two(2) main drawbacks. The first one concerns the correct selection of the number "K" as in the case where it is too small, then great weight is given to the nearest neighbour and in the opposite case, measured neighbours will not be representative of its Class sample since they will be far from the measurement.

** *Euclidean Distance calculation:* $D_i = \sqrt{\sum_{j=1}^{n} \left(RSS_{ij} - RSS_j\right)^2}$

K-Nearest Neighbours Algorithm (KNN)

Here we present the flow of the kNN Algorithm in pseudocode style.

Algorithm 1 kNN Algorithm

Inputs: Data Points **i**, Feature Values **Xi**, Class **Ci**, Euclidean Distance **d**
Output: Let x be a point for which label is not known −− > Find **C** for **X**

Calculate d(x, xi)
  Where i = 1,2,3...n
Sort(d for all n) * non-decreasing order
Take the first k distances from Sort(*list*)
Find those k-points corresponding to the k-distances
Let ki denotes the number of points belonging to the ith class among k points
if ki>kj **then**
  put x in class i
end if

Figure 22. kNN Pseudocode

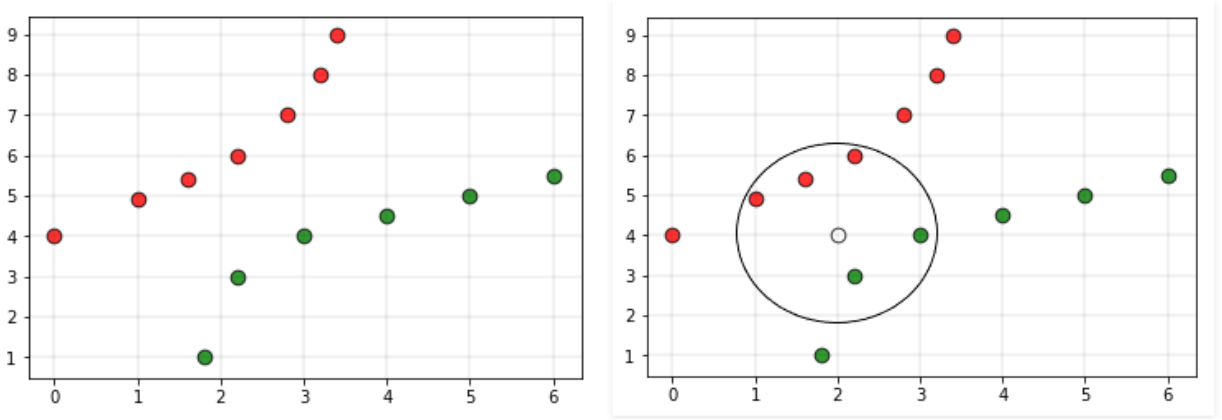
Weighted K-Nearest Neighbours Algorithm (wKNN)

This Algorithm is a modified version of k nearest neighbours. As mentioned earlier, one of the many issues that affect the performance of the kNN algorithm is the choice of the hyper-parameter "k". If it is too small, the algorithm would be more sensitive to outliers. If "k" is too large, then the neighbourhood may include too many points from other classes. In weighted KNN, the nearest "k" points are given a weight using a function called the Kernel Function. The intuition behind wKNN, is to give more weight to the points which are nearby and less weight to the points which are farther away. Any function can be used as a kernel function for the wKNN classifier, whose value decreases as the distance increases. The simple function which is used is the *Inverse Distance Function*. Next an example is presented in order to understand how this Algorithm operates.

Example 1: Considering the following Training Set

Figure 23 (Left): **Data allocation for the example and the point we want to calculate**

Figure 24 (Right): **The points that are selected to define the class which the target point belongs**



The red labels indicate the Class 0 points and the green labels indicate Class 1 points. Consider the White point as the query point *(i.e the point whose class label has to be predicted)*. If we give the above dataset to a KNN based classifier, then the classifier would declare the query point to belong to the class "0". But in the plot, it is clear that the point is closer to the Class 1 points compared to the Class 0 points. *To overcome this disadvantage, wKNN is used.*

wkNN Algorithm

Here we present the flow of the distance wKNN Algorithm in pseudocode style.

Algorithm 1: Weighted kNN Algorithm

Input: Data Points **P**, Number of Nearest Neighbour **k**, Data Table **Dt**
BEGIN
1. distances = Data_{Array}
2. While group in P
3. While feature in p[groups]
§ calculate Distance with Ed
4. Ed = squareroot ((feature[0]-p[0]*2 + feature[1]-p[1]*2)
§ Add Table of Distances
5. distances.add(Ed, groups)
6. distances = sort.distances
7. Choose the 1st **k** distance
8. FREQ_a = 0
§ Weighted Sum of groups[0]
9. FREQ_b = 0
§ Weighted Sum of groups[1]
10. While j in distances
11. if(j[1] == 0)
12. FREQ_a ++, with (1/j[0])
13. Else if (j[1] == 1)
14. FREQ_b ++, with (1/j[0])
15. if(FREQ_a FREQ_b)
16. return 0
17. else
18. return 1
END

Figure 25. wkNN Pseudocode

4.4.2 Proposed Algorithms

In our case, the Algorithm that was used to calculate the room in which a user is, is an implementation of the wKNN Algorithm, which was explained in the previous example *(i.e Example 1)*. Thereafter, the next paragraph describes the process of how our goal is being succeeded by using the wKNN. We first apply the KNN algorithm to our data to find the points with the K closest distances *(i.e K nearest neighbours)*. To implement a W-KNN algorithm there are several functions that calculate weights so that smaller values are considered better than larger values. One of them is the inverse distance function, which calculates the inverse value *(i.e inv_i)* of each distance. Then the sum of these values is calculated as $total\_inv = \sum_{i}^{k} inv_i$.

The Weight of each point *"i"* is defined as $w_i = \frac{inv_i}{total\_inv}$.

To find in which Class *(i.e in which room in our case*) a measurement belongs, the Sum of these Weights is taken per Class they belong. The largest Sum refers to the Class we choose as the most representative for our measurement. An example follows for better understanding of the algorithm used. Suppose we have 9 records in the database and after applying the KNN algorithm

Result after applying KNN algorithm on our data (example)

Point	Room	KNN result
1	1	7
2	1	16
3	2	18
4	2	14
5	2	15
6	3	19
7	4	22
8	4	24
9	4	27

Table 4. Results After kNN

By choosing K=3 *(i.e total points are 9, so K=9=3)*, the three smallest values are taken *(i.e points 1, 4 and 5*). Then, t

Calculation of inverse values and weights on our data (example)

Point	Inverse	Weight
1	0.1429	0.5085
4	0.0714	0.2542
5	0.0667	0.2373

Table 5. Calculation of Weights

Approach 2. Full Grid - Best Match: Our second choice in calculating a user's position in a space, is described next. First, the K-Nearest Neighbours algorithm is used to calculate the 20 points that are closest to the user based on WiFi and mobile radio signals. In fact, the Euclidean distance is used over our data values and the top 20 results are kept, rather than the full KNN algorithm. We recall that the cloud anchors service can compare the image from the device camera with up to 20 stored feature points. For this reason the top 20 are selected. The identifiers of cloud anchors for these signals are returned and the user's device camera is activated to search for them through the cloud anchors service. If at least one point is detected, then we compare the distances of the points that have been detected and we place the user at the nearest cloud anchor that was detected. In this implementation we focus on finding the nearest Cloud Anchor to the user. If no cloud anchor is detected, within the 30 seconds that the user tries to locate one, the user is considered to be at the point that had resulted from the KNN algorithm. In this case it is required to have a cloud anchor at each reference point.

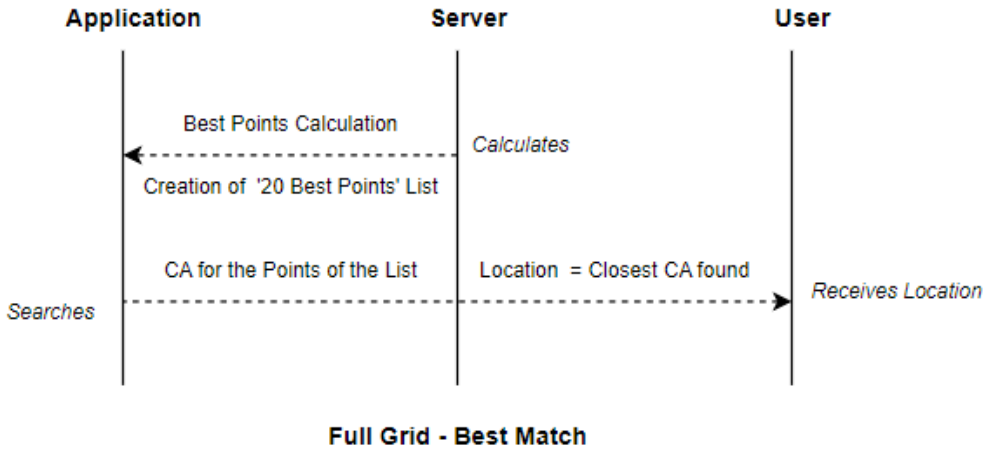


Figure 26. FG/BM Scenario

Approach 3. Partial Grid: In this approach, fewer Cloud Anchors are required to be placed in central points in the indoor space that is mapped. After placing them in some Reference Points, for the rest of our Reference Points the distance can be calculated in relation to the placed Cloud Anchors *(i.e based on the way our grid of points is constructed*). To have a more accurate calculation of the Distance of each Reference Point from the Cloud Anchors, we can additionally measure signals at each point and record the Distance of the point from the Cloud Anchors that have been located *(i.e don't use only the absolute distances resulting from our GRID*). By doing so, Cloud Anchors can be placed in places where there is no predefined point in our GRID and our application will work normally. Locating a user can be done with the same way, as in the previous one. In the first phase, the best 20 points are calculated based on signals. Then, the user locates as many Cloud Anchors are close to him. After a predefined time, for each Cloud Anchor found, the stored Distances recorded for the various points in space along with the measurement of the signals, are used and the KNN algorithm which is applied in these data. Thus, the closest point to the user is obtained, taking into account all the aforementioned data. In this case, it is not required to have one Cloud Anchor per reference point, thereafter the time spent mapping an area and the amount of information used are both significantly reduced.

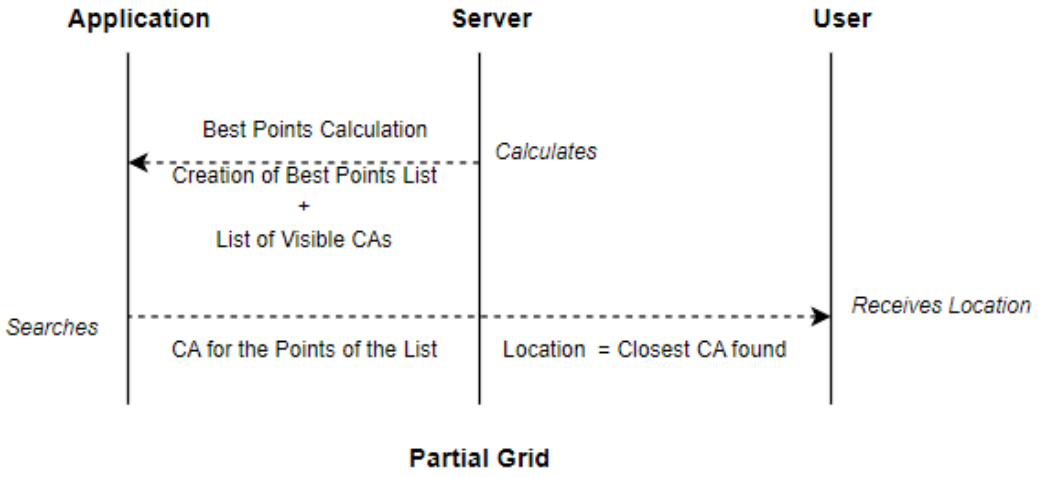


Figure 27. PG Scenario

4.4.3 Matching Scenarios

The subsection here describes a couple of case scenarios, presenting on how our Indoor Positioning System makes decisions, by comparing and matching all the acquired information between a Mapper *(i.e Agent)* and a Matcher *(i.e Client),* in order to define the latter's position in a given space. To this extent, we following scenarios for the Comparison and Matching procedures:
We define some of the users involved in the system *(i.e Agents)*, as *"Trusted"* or *"Untrusted"*. Moreover, we assign *"0"* and *"1"* as the *"Confidence"* values for each one of them, referring to the case where an Agent is considered being a "Trusted" party or the opposite *(i.e Untrusted*). The process is explained next:
Scenario 1: For the "Trusted" Agents we have defined a GRID *(i.e grid is divided into square segmentations of the same size)* on the area's floor, with equal distances *(i.e "d")* between all points of interest. Afterwards, a Training Phase is taking place on the GRID by using Supervised Learning and after obtaining all the necessary measurements *(i.e same amount of data samples)* on each point of interest (i.e position coordinates), the System's Service is passed to the Client.
Scenario 1 - Randomness: Unstructured Process, means that even though a GRID has been defined in the area, still there is a possibility for a user to not be able to obtain all the information needed. This can happen due to many reasons, for example because some values may be missing from the GRID, or the number of the measurements taken in the first place was insufficient. So, at this point we are talking about missing important knowledge about the Positioning System's points of interest or having partial knowledge of it. In this case, the System can make use of a *Randomness* Algorithm, by acquiring all the important data from specific points of interest *(i.e not all points included)* and later trying to define the Client's position on the GRID based on this information.
Scenario 3 - Grid-less: The second idea is having a "Reputation" system. The Indoor Positioning System will be responsible for assigning a *reputation level* on each user, based on his former measurements, with respect to the accuracy level of his provided information. The "good" reputation is gained by storing as much as accurate information in the System *(i.e location of an AP on image, level of accuracy of previous measurements for the AP, etc.)* and a "bad" reputation in any other case. Thereafter, this scenario fits also the previously described one, with the main difference that here we still have Randomness but this time not only with respect to the amount of measurements, but also to the exact positioning of those.

4.4.3.1 Testing Environment

Initially, the area used for testing our proposed model, is an Indoor space of *64m2* with one(1) WiFi Access Point, which is visible from any position in the room. To this extent, *Figure 13* below illustrates the area *(i.e GRID style)* used for our needs.

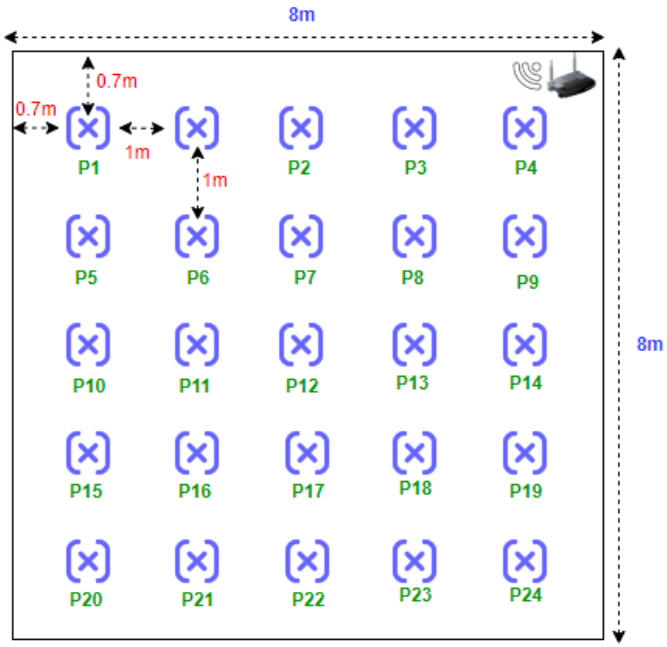


Figure 28. Testing Environment

Moreover, our next goal is to set our testing environment as three(3) different areas *(i.e Residential, Enterprise, Industrial*), in terms of area size and other related factors *(i.e human or machine interference, obstacles etc.),* that could affect our Mapping and Matching procedures. In this case, our proposed model will adapt each time for each case scenario, by taking in account also the information and the conditions applied for each area. The next two(2) Subsections, describe the Mapping and Matching procedures followed in our 3rd Approach *(i.e Partial Grid)* respectively, with the steps followed in each one of them.

4.4.3.2 Mapping & Matching Scenarios

The first step of the Mapping process is the creation of the GRID. We divided our space *(i.e Figure 4.2)* in a GRID having the same distance between its Points of Interest and then we started taking measurements from the available WiFi and Cellular signals. Next step after finishing taking measurements at each Point of Interest, is to place the Cloud Anchors. Following, we move back to the first measured Point and then we calculate the distance between this and each placed Cloud Anchor. Finally, the Mapping is completed when all Points of Interest went through the same process that described in this paragraph.

After the Mapping process is completed, the system passes its service on the Client who wants to locate his position on the GRID. In order to examine the Distances from each Cloud Anchor we followed again the same steps as previously, meaning that we moved at each Point of Interest and then we calculated in detail all the Distances taken from each one. In this step it is necessary to note that, for both processes *(i.e Map and Locate the Points of Interest)* as a final step, we had to normalise our data. This was achieved by dividing each entry *(i.e measurement value)* in our Database, with the equivalent maximum value we can receive for this entry. With respect to the formerly said, for the WiFi Signal measurements we divided with the value *"-120 dbm"* (i.e Signal's Strength maximum value that can be received), for the Cellular Signals with value *"-140 dbm"* and finally for the Visual acquired Data we set as the maximum detection distance to be 10 meters.

4.5 Conclusions

The work presented in this document, describes the Indoor Positioning Systems *(i.e IPS)* that could be used today, for Mapping and Locating someone's position in an Indoor Environment. As mentioned, different kinds of technologies, tools and approaches can be utilised for this purpose, yet with the accuracy of such systems being on top priority when implemented. Thereafter, the Section here will briefly present our tools, technologies and the approaches followed to achieve high levels of accuracy in our IPS. Among others *(i.e Firebase and MySQL Databases for Data storing)*, we implemented an Android Application, where a user can either act as a "Agent" *(i.e. to map an interior space by taking measurements at various points),* or act as a "Client" *(i.e search for his location)*. The Mapping procedure allows the Agent to place an Augmented 3D Object in space marking its location. After defining the room in which he is located along with his coordinates in it, for *30sec* the space is mapped by turning the device camera in all possible angles around that point. At the same time, the device takes four(4) measurements from the available Mobile Radio and WiFi Signals and keeps the average of these values. When *30sec* has passed, he can upload this information by clicking a Button. During the Locating Phase, the device uses information about available Mobile Radio Signals and the available WiFi Networks to locate the closest Reference Points to the user. These reference points are retrieved by the Server and the best 20 are returned sorted to him. Next, the device identifies the image from the device camera with these points, through the Cloud Anchors service. Finally, by using the device's camera and the Augmented Reality capabilities provided by the ARCore, the application calculates the user's location based on the distance from that point and the signals available. Therefore,

the application considers the user's location to be the point that has the best match for the user's data. If no cloud anchor is detected then the user's location is derived only from WiFi and mobile radio signals.

5 MEC service-provisioning for the beyond 5G RAN (CTTC)

5.1 MEC integration with 5G and beyond 5G

5.1.1 Motivation

The four main functions of 5th Generation (5G) are communication, computation, control and content delivery. 5G is not only an advancement in mobile broadband (eMBB) services but it also extends to support vast diversity of massive Machine Type Communication (mMTC) and Ultra-Reliable Low Latency Communication (URLLC) based application services along with support for different verticals. It is foreseen that each application in 5G has different requirements in terms of latency, data rates, reliability/availability, mobility and transmit powers, inducing signification surge in demand for computation and storage resources closer to the users along with enhanced communication resources. 5G alone is not enough for achieving such advancements. The edge computing power of MEC and its integration with 5G brings applications, 3rd party services and content closer to the user in efficient way and is one of the main enablers for turning telecommunication networks into versatile service platforms.

5.1.2 Key enablers for MEC integration with 5G [39]

- 5G Service-Based Architecture (SBA): In 5G core network, the 4th Generation (4G) monolithic Evolved Packet Core (EPC) is disaggregated to implement each function in such a way that it could run independently on a Common Off-The Shelf(COTS) server hardware. It allows 5G core function to be flexible and decentralized
- The ability of an Application Function to influence User Plane Function (UPF) (re)selection and traffic routing directly via the Policy Control Function (PCF) or indirectly via the Network Exposure Function (NEF), depending on the operator's policies
- Local Routing and Traffic Steering: The 5G Core Network selects a UPF close to the UE and executes the traffic steering from the UPF to the local Data Network via a N6 interface. This may be based on the UE's subscription data, UE location, the information from Application Function (AF)
- The Session and Service Continuity (SSC) modes for different UE and application mobility scenarios
- Support of Local Area Data Network (LADN)

5.1.3 MEC integration with 5G [39]

The 5G system architecture specified by 3GPP has been designed with significant changes in RAN and core part to handle wide range of use cases from a massive amount of simple IoT devices to the other

extreme of high bit rate, high reliability mission critical services. The user plane and control plane network functions are worth introducing before diving in to MEC integration with 5G.

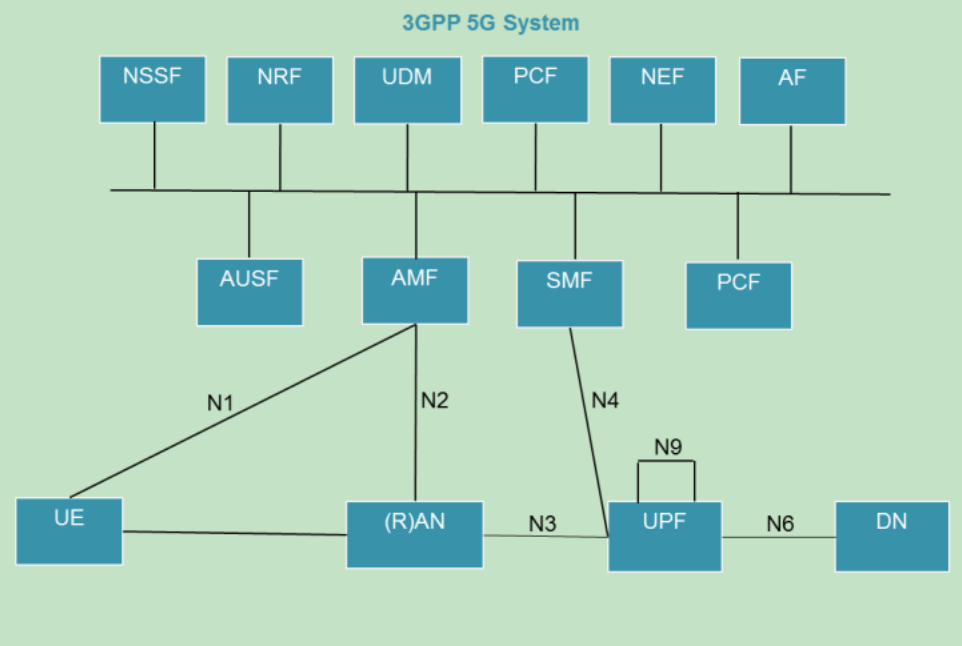


Figure 29. 5G Service-Based Architecture

- Network Resource Function (NRF): Contains the list of network functions and the services they produce
- Network Exposure Function (NEF): Responsible for service exposure and authorization of service access requests
- Application Function (AF): Application influence on traffic routing, accessing NEF and interaction with PCF
- Authentication Server Function (AUSF): Authentication server
- Network Slice Selection Function (NSSF): Responsible for network slice instances selection and allocating necessary AMF for users
- Access and Mobility Management Function (AMF) and Session Management Function (SMF): Handles mobility related procedures, authentication and authorization for the access layer, security anchor functionality and responsible for the termination of RAN control plane and Non-Access Stratum (NAS) procedures, protecting the integrity of signalling, management of registrations, connections and reachability
- Policy Control Function (PCF): Handle's policies and rules of 5G system. It can be accessed directly or via NEF depending on whether the AF is considered trusted or not
- Unified Data Management (UDM): Generates Authentication and Key Agreement (AKA) credentials and responsible for user identification handling, access authorization and subscription management

- User Plane Function (UPF): Distributed and configurable data plane that supports packet routing & forwarding, packet inspection, Quality of Support (QoS) handling, acts as external Protocol Data Unit (PDU) session point of interconnect to Data Network (DN), and is an anchor point for intra- & inter-RAT mobility

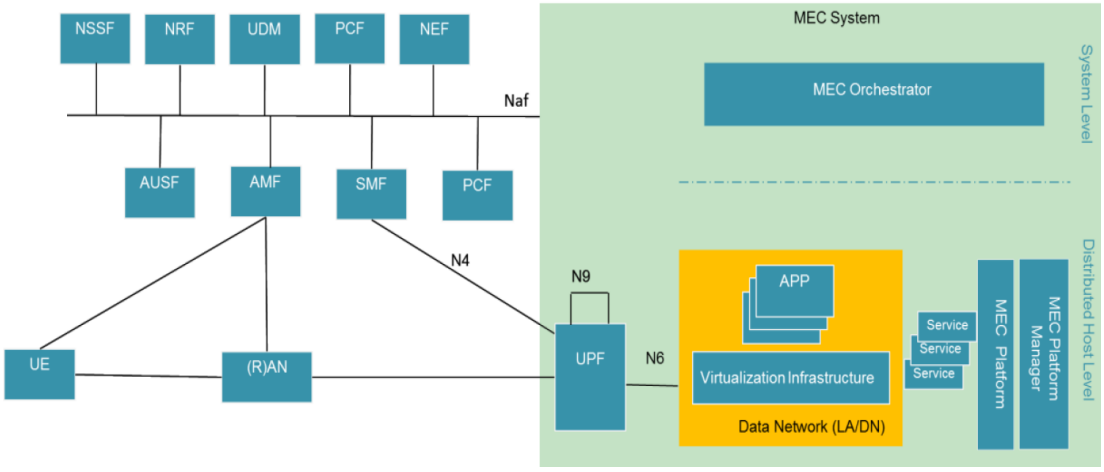


Figure 30. Integrated MEC deployment in 5G network

As shown in Fig. 10, UPF has a major role in MEC integration as it is a distributed and configurable data plane and, in some deployment, locally it can be a part of MEC implementation. MEC orchestrator acts as an AF and interacts with network functions either directly or via NEF. NEF can be deployed centrally together with similar network functions or an instance of NEF can also be deployed in the edge to allow low latency, high throughput service access from a MEC host. MEC host are most often deployed in edge or central data network. UPF steers the user plane traffic towards targeted MEC.

The distributed MEC host accommodates MEC apps, a message broker as a MEC platform service, and another MEC platform service to steer traffic to local accelerators. The choice to run a service as a MEC app or as a platform service is likely to be an implementation choice.

5.1.4 MEC deployment

MEC can be flexibly deployed in different locations depending on requirements [39]. It could be near the Base Station or near the central Data Network. The 4 standard examples are shown in Fig. 11

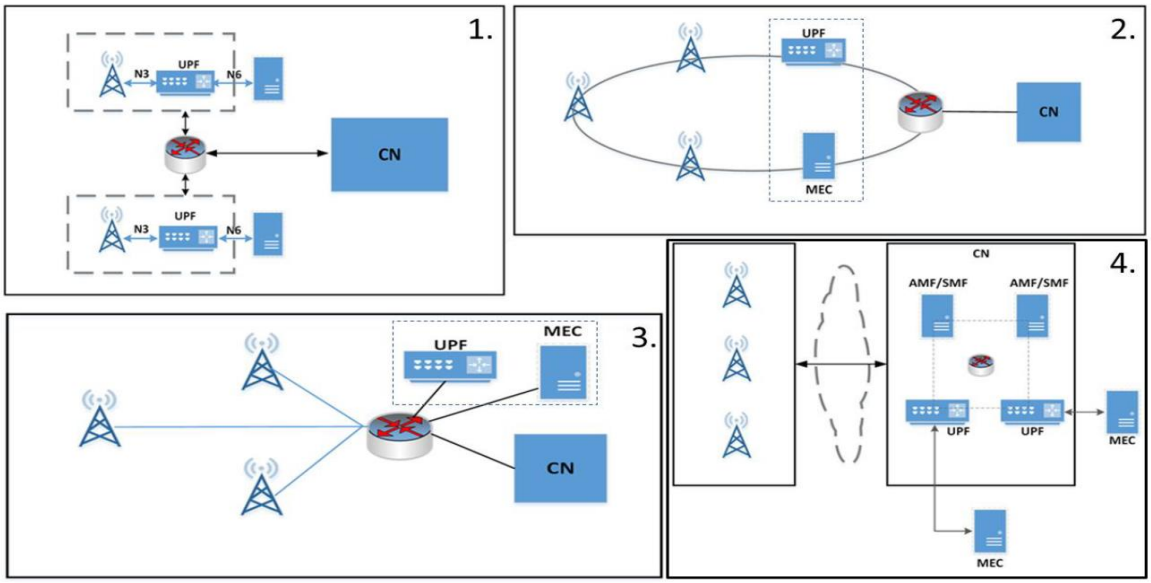


Figure 31. Examples of the physical deployment of MEC

1. MEC and the local UPF collocated with the Base Station
2. MEC collocated with a transmission node, possibly with a local UPF
3. MEC and the local UPF collocated with a network aggregation point
4. MEC collocated with the core network functions

5.2 Open-RAN

5.2.1 Motivation

The traditional wireless network infrastructure makes it difficult for data centers and service providers to tackle massive connections with different services requirements and provide reliable and efficient service to the users. The 5G network helps in solving the above problem with the help of Virtualization, MEC and Network Slicing.

The dependency of RAN part more on hardware and the vendor lock in is causing huge CAPEX and OPEX cost and restricting in building a intelligent cooperative reliable network. So it is important to build 2G, 3G, 4G and 5G RAN solutions based on a general-purpose, vendor-neutral hardware and software-defined technology. Virtualization and RAN disaggregation in 5G helped in building such technology called Open-RAN(O-RAN) whose main principles are openness and intelligence.

5.2.2 Architecture

The main components of ORAN are [40],

- O-RU, O-DU and O-CU whose functionalities are similar to that in 5G disaggregated RAN except with added support of O-RAN based specifications and interface
- Near-Real Time RAN Intelligence Controller (RIC) for control/optimization of RAN elements and resources based on fine grained data using online AI/ML. It is suitable for application with latency needs of between 10ms and 1s
- Non-Real Time RIC for Control/optimization of RAN elements and resources based on coarse grained data using online AI/ML. It is suitable for application with latency requirement greater than 1s. It also provides policy based guidance to near-Real Time RIC.

RIC components can be placed either in edge or core network depending the usage scenario.

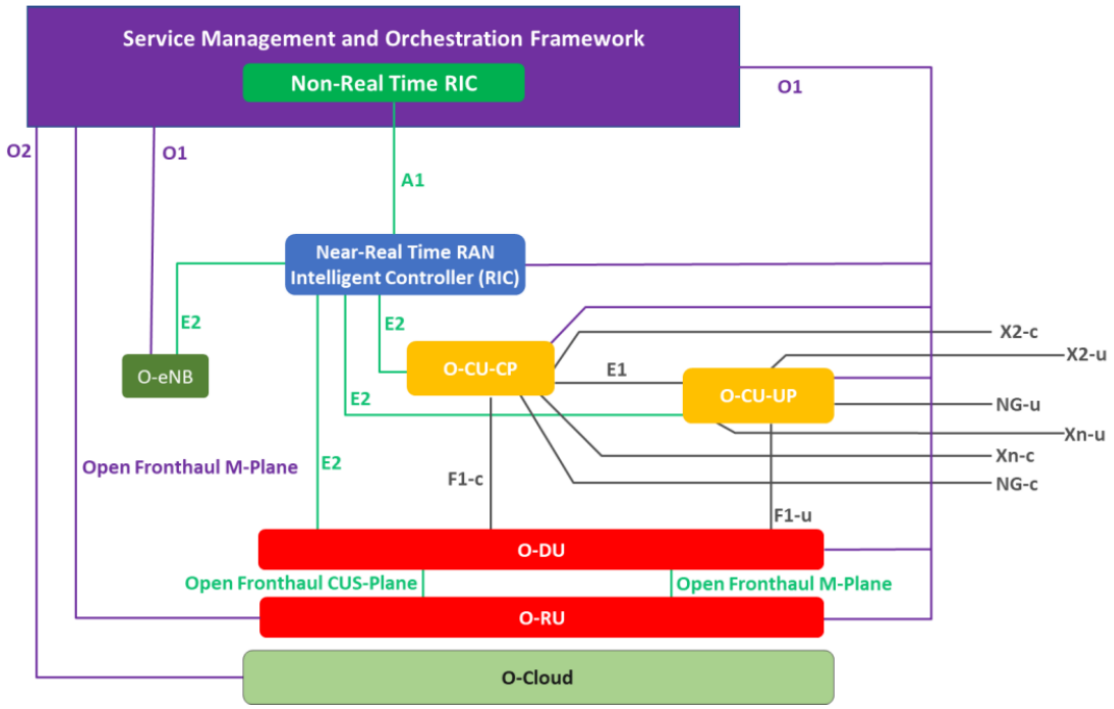


Figure 32. *O-RAN Architecture*

5.3 Cell-free mMIMO

5.3.1 Motivation

Conventional mobile networks (a.k.a. cellular wireless networks) are based on cellular topologies where a land area is divided into cells and each cell is served by one base station. Future wireless networks are expected to manage to serve billions of devices simultaneously with high throughput demands for supporting many applications like voice, real-time video, high quality movies, etc. Cellular networks fail to handle such huge connections efficiently since user terminals at the cell boundary suffer from very high

interference, and hence, perform badly. Furthermore, conventional cellular systems are designed mainly for human users. In future wireless networks, machine-type communications such as the Internet of Things, Internet of Everything, Smart X and many more are expected to play an important role. The main challenge of machine-type communications is scalable and efficient connectivity for billions of devices. Centralized technology with cellular topologies does not seem to be working for such scenarios since each cell can cover only a limited number of user terminals [43].

The combination between cell-free structure and massive MIMO(mMIMO) technology yields to the new concept: Cell-free mMIMO. Cell-free mMIMO network infrastructure is a beneficial epitome of the general distributed massive MIMO concept. By relying on time-division duplex (TDD) operation, a large number of geographically distributed antennas jointly serve a lower number of UEs with the aid of a fronthaul network and a central processing unit (CPU) operating with same time-frequency resource. The cellular or cell boundary concepts disappear in cell-free mMIMO. The basic premise behind cell-free systems is to reap all the benefits of network MIMO solutions by providing many more antennas than the number of users which allows us to invoke transmit pre-coding (TPC) for eliminating the interference at the UEs. The key properties of massive MIMO can be beneficially exploited for supporting scalable implementations. Then, the noise, fading and inter-user interference can be averaged out by relying on the law of large numbers. As a benefit, cell-free systems are capable of providing better service than conventional uncoordinated small cells [41].

5.3.2 Architecture

Cell-free mMIMO consist of many geographically distributed Access Points (APs) with single/multiple antennas jointly serving a lower number of User Equipment's (UEs) with the aid of a fronthaul links and a Central Processing Unit (CPU) operating at the same time-frequency resource. The APs do not exchange any Channel State Information´s (CSI) and the coordination and power control between APs are handled by the CPU. Each AP of CF massive MIMO systems only requires local CSI to perform both transmit preprocessing and signal detection. It is beneficial for low latency machine critical applications.

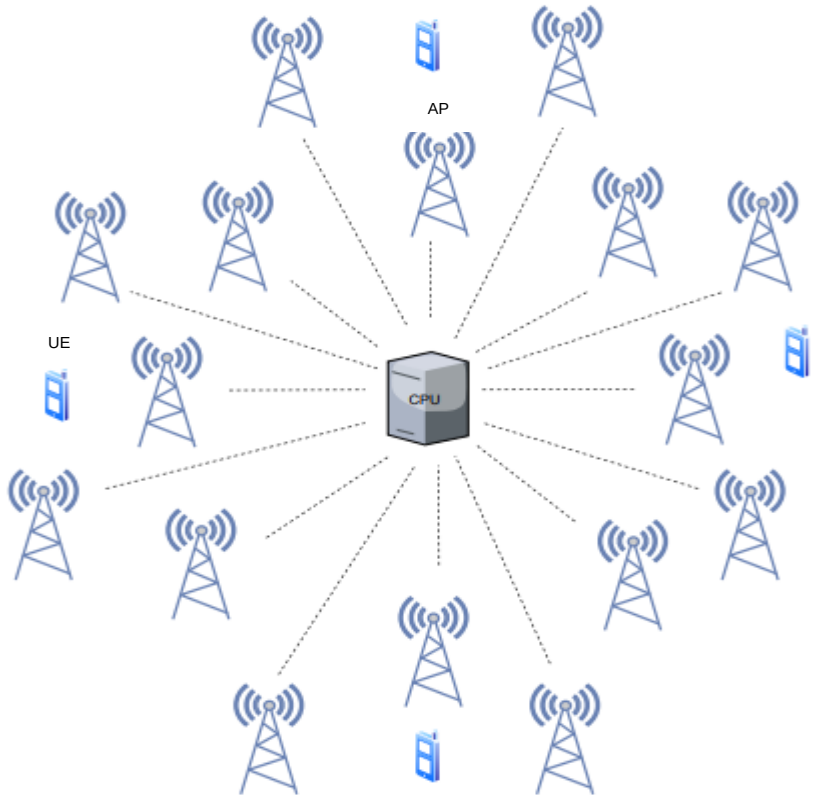


Figure 33. *Cell-free mMIMO system [42]*

5.4 ANN aided joint APs selection & beamforming computation for cell-free mMIMO O-RAN system

5.4.1 Cell-free mMIMO O-RAN system

The hardware and software supporting the O-RAN disaggrecgation concept can support the needs of cell-free mMIMO with fine granularity of the processing options(processing at AP vs CPU, Inter-CPU coordination). The CPU, which is one of the key components of cell-free mMIMO, is dis-aggregated into O-CU and O-DU. The APs are considered to be O-RU. Among many split options, distribution of RAN functions between the above parts of network is by function split 8 or split 7.2x, as per 3rd Generation Partnership Project (3GPP) specification. In Split 8, O-RUs are responsible only for converting signals while even the beamforming is shifted to the O-DU. This split requires very high fronthaul capacity. On the other hand, it can be attractive for a network where a vast number of (low price) O-RUs have to be deployed. In Split 7.2x, O-DUs are responsible for signal encoding and decoding, while O-RUs perform some light-weight processing. They can be used for centralized CF mMIMO networks while having lower fronthaul requirements. These split options make CPU processing easier and helps in maintaining balance between fronthaul capacity and O-RU complexity depending on local processing to be done [43]. The RIC helps in

intelligent service management and integration of 3rd party services. The AI/ML based dynamic control mechanism and any optimization algorithms can be installed in RIC. This opens up multiple problem areas related to power control, efficiency improvement, flexible localization, resource allocations, AP selection, clustering, reducing latency and load balancing.

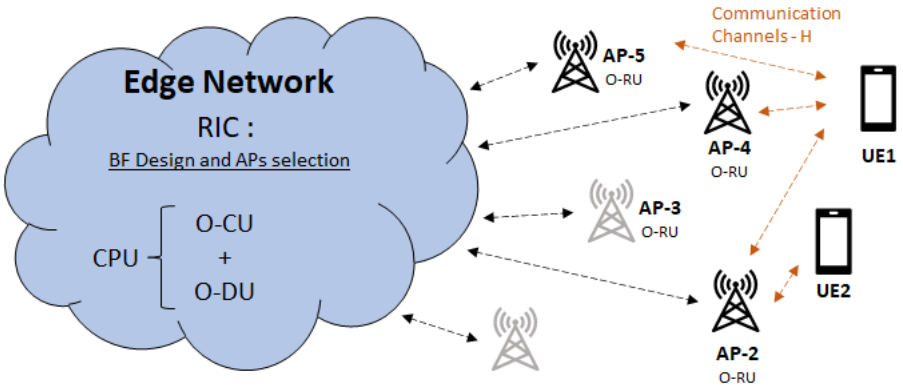


Figure 34. *Cell-free mMIMO O-RAN system*

5.4.2 Motivation

High spectral efficiency in cell-free is mainly due to the joint service offered by large number of distributed APs. Aligned with the Green Deal initiative promoted by the European Union and in line with current 5G/Beyond 5G(B5G) standards, energy efficiency is the fundamental metric in the forthcoming wireless generation. While power consumption in mMIMO mostly addressed issues on radiofrequency and baseband processing units, cell-free mMIMO must consider new energy-related aspects such as the presence of a dedicated fronthaul connection linking the APs to the CPU and the fixed power expenditure each individual AP entails. These fixed power terms are one of the majorly contributing metric to the total energy expenditure making the consideration of activation of only few APs a wise option.

Large number of densely deployed APs causes interference and fading. Rapid changes in fading and high computation complexity causes difficulty in optimization of non-convex power control methods as they are np-hard. Load traffic conditions vary over space and time and some APs may be underutilized at certain time periods and only contribute to the increased overall energy consumption of the network. If an AP has negligible contribution to the system performance, it is beneficial to turn off that AP to save energy. Most research on cell-free focuses on problems like just beamforming design, energy efficiency or AP selection and clustering separately, addressing issues with simple setup like single CPU and few APs. Different AP switch on/off strategies were presented in [25], where the focus was on suboptimal heuristic algorithms to reduce the complexity when solving the NP-hard AP selection problem. The previous works still relied either on heuristics or conventional optimization methods [45]. This motivates to investigate the

joint design of intelligent AP activation and beamforming design to further improve the energy usage in CF-MIMO.

5.4.3 Objective

The main aspects of proposed framework are summarized as follows,

- Joint beamforming design and APs selection
- Provide satisfied QoS to users
- Intelligent and scalable optimization algorithm
- Minimize power consumption
- Reduce data overhead in the network

5.4.4 Methodology

A cell-free mMIMO communication system is considered in which N APs coherently serve K users/UEs that are randomly distributed on the coverage area. Each AP is equipped with M antennas, while the users are assumed to have a single antenna. A CPU connects with the APs via a fronthaul network. We aim to minimize the power consumption, while guaranteeing certain minimum QoS represented as γ_o (in terms of SINR) for each user via joint beamforming design and AP subset (b_n) selection from total available APs N. The APs energy consumption includes the energy of electromagnetic radiation, the energy consumed in hardware components and the energy consumed in the backhaul. If there are large number of densely deployed APs, the AP that has negligible contribution to the system performance can be avoided to serve the UE by selecting a subset of APs that contributes significantly to user performance, thereby enabling the cell-free network to provide good service while achieving power savings and reduced fronthaul load.

The mathematical formulation of the problem is Mixed Integer Nonlinear Programming(MINLP) as shown below, which is NP-hard. It can be reformulated as a non-convex Quadratically Constrained Quadratic Problem(QCQP). The minimization problem can be efficiently solved by Successive Convex Approximation(SCA), Alternate Direction Method of Multipliers(ADMM) and other statistical methods that provides fast computation, distributed optimization and parallel processing.

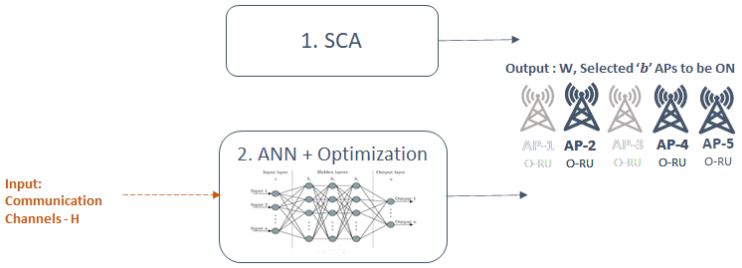


Figure 35. *Joint beamforming design and APs selection*

$$\min_{\{w_k\}_{k=1}^{K},\{b_n\}_{n=1}^{N}} \frac{1}{\eta}\sum_{k=1}^{K}||w_k||^2 + P_{FIX}\sum_{n=1}^{N} b_n$$

$$subject\ to,$$

$$\sum_{k=1}^{K}||w_k^{[n]}||^2 < P_n \qquad n = 1,2,\dots N$$

$$SINR_k \geq \gamma_o \qquad k = 1,2,\dots K$$

$$b_n \in \{0,1\} \qquad n = 1,2,\dots N$$

Here, P_{FIX} is the signaling power of AP plus power consumed by the circuit and P_n is the maximum power available at each AP. As of now, the optimization problem is solved in 2 ways. First, with the help of SCA optimization algorithm and second, using Artificial Neural Network(ANN) + Optimization. Second method is a 2-stage procedure performing AI based AP subset b_n selection where ANN is trained to give best set of APs to be ON based on channel and reduced problem optimization for computation of beamforming weights w_k.

5.4.5 Results

The effectiveness of the proposed joint design is shown via numerical results in terms of total transmit power and computation time changing with respect to number of APs ON. For computation 6 APs each with 4 antennas are considered to available in the environment along with 4 UEs.

Fig. 16 compares the total transmit power achieved by proposed design(orange bar) where only an appropriate APs are selected to be active with the fact when all the APs are considered to be active(blue bar). The result for proposed proposed designs SCA and ANN + optimization is same. It is evident that the total transmit power required is very high when all the APs are ON. When the number of APs are increased,

the transmit power is also seen to be increasing incase when all APs are active. In case of proposed design increase in number of APs has no much effect on transmit power.

Fig. 17 compares the computation complexity of both proposed algorithms i.e. SCA and ANN + optimization. It can be seen that, the SCA takes more time to jointly select APs and compute BF parameters and also with the increase in number of APs the computation complexity also increases. In case of ANN + optimization, the computation complexity remains same for any number of APs and also faster than SCA.

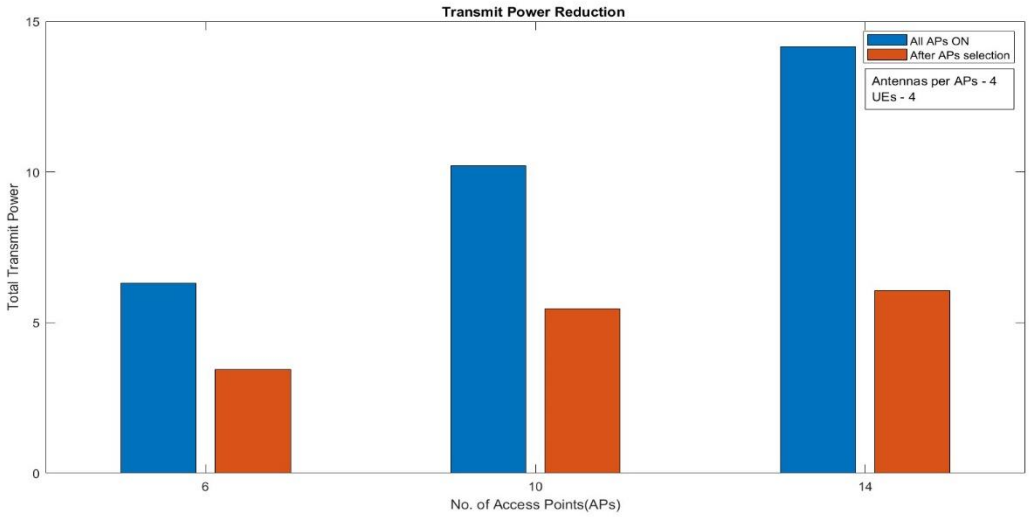


Figure 36. *Total transmit power vs Number of APs*

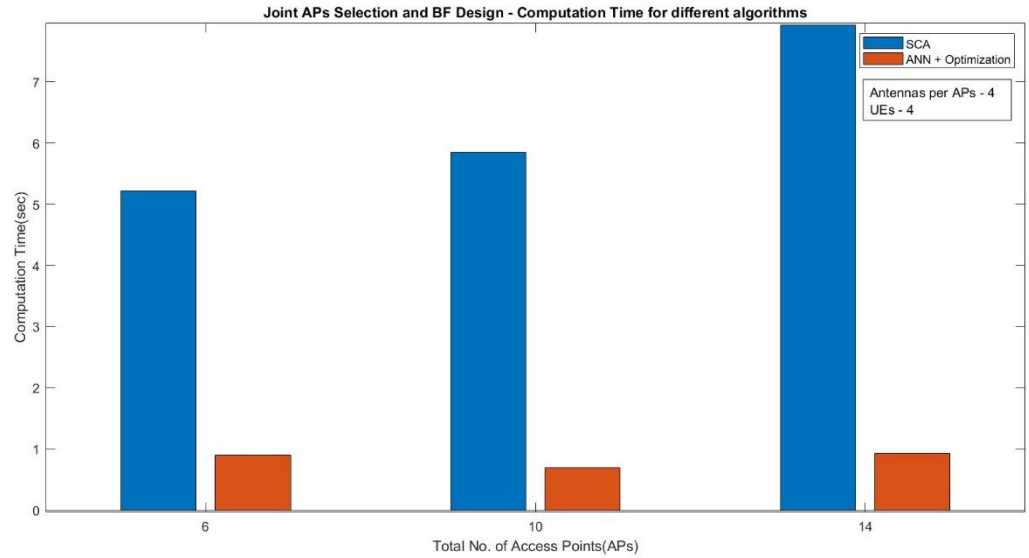


Figure 37. *Computation time vs Number of APs*

5.4.6 Conclusion

Power efficiency is achieved by selecting proper subset of APs with the help of 2 proposed designs SCA and ANN + Optimization. AI-enhanced subset selection provides reduced computation time, but more fine-tuning of ANN is to be done. As a future work, Other mathematical tools like ADMM, deep reinforcement learning, branch and bound, etc., will be considered for comparison.

5.5 XAI and Provisioning of resources in O-RAN

5.5.1 XAI

Since several years, Deep learning (DL) methods are state of the art models for problems with time series as input data. Recurrent methods are adapted to work with time series because of their memory state and their ability to learn relations through time. Convolutional neural networks (CNNs) with temporal convolutional layers are able to build temporal relationships and generate high level features from raw data. The introduction of these methods to work on time series helps in increasing model accuracy and avoids the heavy data pre-processing. However, the major drawbacks of these methods are the lack of interpretability due to their high complexity and uncertainty. Predictions are not only expected to provide us the value that, according to the model, is most likely going to be close to the ground truth, but also a measure for how certain we can be about that value. EXplainable Artificial Intelligence (XAI) is therefore a key concern for time series as most state-of-the-art methods are not interpretable [46].

Probabilistic forecasting summarizes what is known about future events. In contrast to single-valued, probabilistic forecasts assign a probability to each of the number of different outcomes, and the complete set of probabilities represents a probability forecast. It estimates a joint distribution over future values of a time series, given a sequence of historical values. As an important problem with many applications and an abundance of real-world data, probabilistic forecasting has unsurprisingly come to be dominated by deep learning methods.

5.5.2 Objective and Literature survey

The prediction of the probabilities of the KPIs and the time series of network resource usage in O-RAN must be considered for better resource provisioning in 6G systems.

Literature [47] shows that Long Short-Term Memory (LSTM) networks out-perform other machine learning approaches for time-series analysis in the traffic prediction. LSTM structures have been proposed and datasets consist of the spatio-temporal distribution of the mobile traffic in different base stations. Spatial correlation has been used to evidence similarities between neighbouring base stations. Even though LSTM has been applied for the traffic prediction, the input data consider only one metric (e.g. the traffic,

spatially distributed), and only a one-step prediction is considered which makes it uncertain for real-world cases.

According to literature [48], the slices are generated not only for different kinds of services but also for different tenants running in parallel in the networks. Compared with the conventional scenario, the demand for slicing in the vertical industry is more diversified and customized. Taking smart grid as an example, different tenant networks are featured with diverse service level agreements (SLAs), such as load control, distribution automation, sensor meter, inspection, operation, etc. For each tenant network, there will be multiple network slices belonging to different management roles to realize similar services, a large number of network resources will be consumed if these network slices are completely isolated. In fact, for some services such as inspection and operation, their traffic is elastic and the strict resource isolation between the slices is not necessary, the basic functions can be achieved with a certain amount of dedicated resources guaranteed, and the advanced functions can also be realized with more resources. Traffic uncertainty has been an impediment to the on-demand allocation of slicing resources. Paper compares different prediction methods and demonstrates that a better prediction result can reduce the uncertainty of the traffic. Researchers have widely adopted a prediction-based approach to dynamically adjusting network slicing resources to fit real-time changing traffic. Lots of predictive tools such as machine learning, deep neural network, naive benchmark, holt-winters, etc., are proposed to realize the implementation of proactive dynamic network slicing based on prediction. However, the prediction error always exists regardless of the prediction method, which leads to over-provisioning or under-provisioning causing resource waste or service degradation. More accurate prediction methods without uncertainty are preferred in order to perform efficient dynamic allocation of resources.

6 Conclusions and future plans (one paragraph per ESR)

In chapter 2, we first addressed MS placement by proposing a low-complexity heuristic iplace that minimizes the number of used servers while meeting the performance requirements of the MSs by placing the MSs that interfere with each other on different servers. Secondly, we characterized various container states used in literature by measuring their memory and cpu utilization to minimize start-up latencies of the MSs in serverless computing scenarios. Our future work will leverage the characteristics of various container states proposed in the literature and propose an efficient container retention strategy for the edge nodes that minimizes the average cold-start latency while efficiently utilizing the limited edge resources.

In chapter 3, we presented related work in content placement and delivery in MEC-empowered Networks, which includes QoE in ICNs, optimization in content caching and PoC implementation. Also, our proposal for the special case of more than one single intermediate MEC-server which between the CDN and the user requesting the content, is presented. We focus on the scenario of known cached content and links' average throughput (both CDN-to-BS and BS-to-user), formulating the selection problem of video segments' delivery to the end user without stalling and getting the exact sequence of the different MEC-enabled proxy-servers that deliver each segment. In our future work we will propose algorithms for the formulated problem while we will also focus on the content caching methods.

In Chapter 4, we described in analysis a model for Indoor Positioning Systems implementations. Our proposed model is based on the Fingerprint technique by taking advantage the Augmented Reality technology in combination with measurements from both WiFi and Cellular Access Points. Even though our preliminary results has not been included in this document yet, still our simulations so far have shown a great increase in our system's accuracy *(i.e user's positioning in Indoor space)*. Our next plans include additional implementations and simulations of our proposed model, for different kinds of Indoor Environments *(i.e Industrial, Enterprise, Residential)*, with some of those scenarios to have already been described in Subsection 4.4.3.

In chapter 5, joint design for AP activation and beamforming design based on SCA and ANN to minimize the total power consumption while satisfying the predefined QoS requirements as well as per-AP transmit power constraint is proposed. It is shown that the proposed design significantly reduces the total power consumption by selecting proper subset of APs, compared with the all activated AP. AI-enhanced subset selection provides reduced computation time, but still the ANN is to be fine-tuned for future work. Joint beamforming design and AP selection will be implemented using other mathematical tools for comparison (ADMM, deep reinforcement learning, Branch&Bound, etc.) and work on a conference paper and a journal will be started. Research on XAI and provisioning of resources in O-RAN would be done parallelly.

7 References

[1] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, “A survey on service function chaining,” J. of Network and Computer Applications, 2016.

[2] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. Ramakrishnan, and T. Wood, “OpenNetVM: A platform for high performance network service chains,” in ACM HotMiddlebox, 2016.

[3] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, “ClickOS and the art of network function virtualization,” in USENIX NSDI, 2014.

[4] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, “NetBricks: Taking the V out of NFV,” in ACM OSDI, 2016.

[5] S. Palkar and et al., “E2: A Framework for NFV Applications,” in ACM SOSP, 2015.

[6] A. Manousis, R. A. Sharma, V. Sekar, and J. Sherry, “Contention-aware performance prediction for virtualized network functions,” in ACM SIGCOMM, 2020.

[7] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa, “Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations,” in IEEE/ACM MICRO, 2011.

[8] M. Dobrescu, K. Argyraki, and S. Ratnasamy, “Toward predictable performance in software packetprocessing platforms,” in ACM NSDI, 2012.

[9] A. Tootoonchian, A. Panda, C. Lan, M. Walls, K. Argyraki, S. Ratnasamy, and S. Shenker, “ResQ: Enabling SLOs in network function virtualization,” in USENIX NSDI, 2018.

[10] V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, and M. M. Swift, “Resource-freeing attacks: Improve your cloud performance (at your neighbor’s expense),” in ACM CCS, 2021.

[11] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, “Demystifying the performance interference of co-located virtual network functions,” in IEEE INFOCOM, 2018.

[12] Q. Zhang, F. Liu, and C. Zeng, “Adaptive interference-aware VNF placement for service-customized 5G network slices,” in IEEE INFOCOM, 2019.

[13] Q. Zhang, F. Liu, and C. Zeng, “Online adaptive interference-aware VNF deployment and migration for 5G network slice,” IEEE/ACM Transactions on Networking, 2021.

[14] Y. Mu, L. Wang, and J. Zhao, “Energy-efficient and interference-aware vnf placement with deep reinforcement learning,” in 2021 IFIP, 2021.

[15] D. Novakovic, N. Vasi ´ c, S. Novakovi ´ c, D. Kosti ´ c, and R. Bianchini, “DeepDive: Transparently identifying and managing performance interference in virtualized environments,” in USENIX ATC, 2013.

[16] S. Song, C. Lee, H. Cho, G. Lim, and J.-M. Chung, “Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks,” IEEE Transactions on Mobile Computing, 2020.

[17] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” IEEE Transactions on Information Theory, 1975

[18] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, “Cloud programming simplified: A berkeley view on serverless computing,” 2019. [Online]. Available: http://arxiv.org/abs/1902.03383

[19] M. Shahrad, R. Fonseca, ´Inigo Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, “Serverless in the ˜ wild: Characterizing and optimizing the serverless workload at a large cloud provider,” in USENIX ATC, 2020.

[20] P. Silva, D. Fireman, and T. E. Pereira, “Prebaking functions to warm the serverless cold start,” in ACM Middleware, 2020, pp. 1–13.

[21] A. Mohan, H. Sane, K. Doshi, S. Edupuganti, N. Nayak, and V. Sukhomlinov, “Agile cold starts for scalable serverless,” in USENIX HotCloud, 2019.
[22] H. D. Nguyen, Z. Yang, and A. A. Chien, “Motivating high performance serverless workloads,” in ACM HiPS, 2021, p. 25–32.
[23] L. Pan, L. Wang, S. Chen, and F. Liu, “Retention-aware container caching for serverless edge computing,” in IEEE INFOCOM, 2022, pp. 1069–1078.
[24] “Squeezing the milliseconds: How to make serverless platforms blazing fast!” Squeezing the milliseconds: How to make serverless platforms blazing fast! | by Markus Thömmes | Apache OpenWhisk | Medium.
[25] https://openwhisk.apache.org/
[26] T. M. Ayenew, D. Xenakis, N. Passas and L. Merakos, "Cooperative Content Caching in MEC-Enabled Heterogeneous Cellular Networks," in IEEE Access, vol. 9, pp. 98883-98903, 2021, doi: 10.1109/ACCESS.2021.3095356.
[27] W. Li, S. M. A. Oteafy, M. Fayed and H. S. Hassanein, "Quality of Experience in ICN: Keep Your Low-Bitrate Close and High-Bitrate Closer," in IEEE/ACM Transactions on Networking, vol. 29, no. 2, pp. 557-570, April 2021, doi: 10.1109/TNET.2020.3044995.
[28] X. Huang, L. He, L. Wang and F. Li, "Towards 5G: Joint Optimization of Video Segment Caching, Transcoding and Resource Allocation for Adaptive Video Streaming in a Multi-Access Edge Computing Network," in IEEE Transactions on Vehicular Technology, vol. 70
[29] S. -R. Yang, Y. -J. Tseng, C. -C. Huang and W. -C. Lin, "Multi-Access Edge Computing Enhanced Video Streaming: Proof-of-Concept Implementation and Prediction/QoE Models," in IEEE Transactions on Vehicular Technology, vol. 68, no. 2, pp. 1888-1902, Feb. 20
[30] A. Mehrabi, M. Siekkinen and A. Ylä-Jääski, "Edge Computing Assisted Adaptive Mobile Video Streaming," in IEEE Transactions on Mobile Computing, vol. 18, no. 4, pp. 787-800, 1 April 2019, doi: 10.1109/TMC.2018.2850026.
[31] Xu, He, Ye Ding, Peng Li, Ruchuan Wang, and Yizhu Li. 2017. "An RFID Indoor Positioning Algorithm Based on Bayesian Probability and K-Nearest Neighbor" Sensors 17, no. 8: 1806. https://doi.org/10.3390/s17081806
[32] Xingbin Ge and Zhiyi Qu, "Optimization WIFI indoor positioning KNN algorithm location-based fingerprint," 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016, pp. 135-137, doi: 10.1109/ICSESS.2016.7883033
[33] Z. Liu, X. Luo and T. He, "Indoor positioning system based on the improved W-KNN algorithm," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017, pp. 1355-1359, doi: 10.1109/IAEAC.2017.8054235
[34] Dai, Peng & Yang, Yuan & Wang, Manyi & Yan, Ruqiang. (2019). Combination of DNN and Improved KNN for Indoor Location Fingerprinting. Wireless Communications and Mobile Computing. 2019. 1-9. 10.1155/2019/4283857
[35] Yongliang Sun, Yubin Xu, Lin Ma and Zhian Deng, "KNN-FCM hybrid algorithm for indoor location in WLAN," 2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS), 2009, pp. 251-254, doi: 10.1109/PEITS.2009.5406793.
[36] Ferreira, David, Richard Souza, and Celso Carvalho. 2020. "QA-kNN: Indoor Localization Based on Quartile Analysis and the kNN Classifier for Wireless Networks" Sensors 20, no. 17: 4714. https://doi.org/10.3390/s20174714
[37] H. Gan, M. H. B. M. Khir, G. Witjaksono Bin Djaswadi and N. Ramli, "A Hybrid Model Based on Constraint OSELM, Adaptive Weighted SRC and KNN for Large-Scale Indoor Localization," in IEEE Access, vol. 7, pp. 6971-6989, 2019, doi: 10.1109/ACCESS.2018.2890111.
[38] Guo, T., Chai, M., Xiao, J. et al. A Hybrid Indoor Positioning Algorithm for Cellular and Wi-Fi Networks. Arab J Sci Eng 47, 2909–2923 (2022). https://doi.org/10.1007/s13369-021-05925-9
[39] ETSI White paper No.28., “MEC in 5G networks,” ISBN No. 979-10-92620-22-1, First edition June 2018.

[40] O-RAN Alliance White paper, “O-RAN use cases and deployment scenarios, Towards open and smart RAN,”, February 2020

[41] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson and T. L. Marzetta, "Cell-Free Massive MIMO Versus Small Cells," in *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1834-1850, March 2017

[42] W. A. Chamalee Wickrama Arachchi, K. B. Shashika Manosha, N. Rajatheva, M. Latva-aho, "On Max-Min SINR with MMSE Processing for Uplink Cell-Free Massive MIMO," [online] Available: *arxiv.org/pdf/1908.03187v1.pdf*, August 2019

[43] V. Ranjbar, A. Girycki, M. A. Rahman, S. Pollin, M. Moonen and E. Vinogradov, "Cell-Free mMIMO Support in the O-RAN Architecture: A PHY Layer Perspective for 5G and Beyond Networks," in *IEEE Communications Standards Magazine*, March 2022

[44] C. F. Mendoza, S. Schwarz and M. Rupp, "Deep Reinforcement Learning for Dynamic Access Point Activation in Cell-Free MIMO Networks," *WSA 2021; 25th International ITG Workshop on Smart Antennas*, 2021

[45] T. X. Vu, S. Chatzinotas, S. ShahbazPanahi and B. Ottersten, "Joint Power Allocation and Access Point Selection for Cell-free Massive MIMO," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020

[46] Thomas Rojat, Raphael Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Diaz-Rodriguez, “Explainable Artificial Intelligence(XAI) on Time Series Data: A Survey,” [online] Available: arxiv.org/pdf/2104.00950v1.pdf, April 2021

[47] H. D. Trinh, L. Giupponi and P. Dini, "Mobile Traffic Prediction from Raw Data Using LSTM Networks," *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018

[48] Q. Guo, R. Gu, H. Yu, T. Taleb and Y. Ji, "Probabilistic-Assured Resource Provisioning with Customizable Hybrid Isolation for Vertical Industrial Slicing," in *IEEE Transactions on Network and Service Management*, 2022

[49] GluonTS – Probabilistic Time Series Modeling in Python, [online] Available: ts.gluton.ai/stable

List of Acronyms and Abbreviations

Acronym	*Description*
3GPP	Third Generation Partnership Project
5G	5th generation (5G) mobile network
ABR	Adaptive Bitrate
AF	Application Function
AI/ML	Artificial Intelligence / Machine Learning
AMF	Access and Mobility Management Function
AP	Access Point
AUSF	Authentication Server Function
BBU	Baseband Unit
BFT	Byzantine Faulty Tolerant
biRNN	Bidirectional Recurrent Neural Network
BPP	Binomial Point Process
BS	Base Station
CAT	Cache Allocation Technology
CDF	Cumulative Distribution Function
CDN	Content Distribution Network
CFS	Customer Facing Service
CGI	Global Cell Identity
CHR	Cache Hit Ratio
CNN	Convolutional Neural Network
COTS	Common Off-The Shelf
CP	Content Provider
CRAN	Cloud Radio Access Network
CRF	Combined Recency and Frequency
CS	Content Server
D2D	Device-to-Device Communication
DASH	Dynamic adaptive streaming over HTTP
DDIO	Data Direct I/O
DN	Data Network
DNS	Domain Name System
DP	Dynamic Programming
DPDK	Data Plane Development Kit
EAS	Edge Application Server
EEC	Edge Enabler Client
EES	Edge Enabler Server
eMBB	Enhanced Mobile Broadband
EPC	Evolved Packet Core
ESN	Echo State Network
ETSI	European Telecommunications Standards Institute
EWMA	Exponential Weighted Moving Average
FBS	Femto Base Station
FIFO	First In First Out
FL	Federated Learning
FNN	Feedforward Neural Network
HAS	HTTP Adaptive Streaming
HCPP	Hard Core Point Process
HHM	Hidden Markov Model
IMEI	International Mobile Equipment Identity
KPI	Key Performance Indicator

LADN	Local Area Data Network
LFU	Least Frequently Used
LN	Lightning Network
LRFU	Least Recently/Frequently Used
LRU	Least Recently Used
LSTM	Long Short-Term Memory
MBA	Memory Bandwidth Allocation
MBS	Macro Base Station
MDF	Media Description File
MDP	Markov Decision Process
ME	Mobile Equipment
MEC	Multi-Access Edge Computing
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MLP	Multilayer Perceptron
mMTC	massive Machine Type Communication
MNO	Mobile Network Operator
MPD	Model Predictive Control
NAS	Non-Access Stratum
NEF	Network Exposure Function
NFV	Network Function Virtualization
NLP	Neural Language Processing
NN	Neural Network
NRF	Network Resource Function
NSSF	Network Slice Selection Function
O-RAN	Open-RAN
OSS	Operations Support System
OTT	Over-the-Top
PC	Payment Channel
PCF	Policy Control Function
PCP	Poisson Cluster Process
PDU	Protocol Data Unit
PoA	Proof-of-Authority
PoC	Proof-of-Concept
PoS	Proof-of-Stake
PoW	Proof-of-Work
PPP	Poisson Point Process
QoE	Quality of Experience
QOS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RDT	Resource Director Technology
ReLU	Rectified Linear Unit
RIC	RAN Intelligence Controller
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RRH	Remote Radio Head
RTT	Round Trip Time
RWP	Random Waypoint
SBA	Service-Based Architecture
SBS	Secondary Base Station
SC	Smart Contract

SCN	Small Cell Network
SINR	Signal to Interference Noise Ratio
SSC	Session and Service Continuity
TPS	Transactions per Second
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-Reliable Low Latency Communication
VL	Virtual Link
VNF	Virtual Network Function
VNFFG	Virtual Network Function Forwarding Graph
VoD	Video on Demand