Scheduling Latency-Sensitive Applications in Edge Computing

Vincenzo Scoca[1], Atakan Aral[2], Ivona Brandic[2], Rocco De Nicola[1] and Rafael Brundo Uriarte[1]
[1]*IMT School for Advanced Studies Lucca, Italy*
[2]*Vienna University of Technology, Austria*

Keywords: Edge Computing, Scheduling, Latency-Sensitive Services, Live Video Streaming, Resource Selection.

Abstract: Edge computing is an emerging technology that aims to include latency-sensitive and data-intensive applications such as mobile or IoT services, into the cloud ecosystem by placing computational resources at the edge of the network. Close proximity to producers and consumers of data brings significant benefits in latency and bandwidth. However, edge resources are, by definition, limited in comparison to cloud counterparts, thus, a trade-off exists between deploying a service closest to its users and avoiding resource overload. We propose a score-based edge service scheduling algorithm that evaluates both network and computational capabilities of edge nodes and outputs the maximum scoring mapping between services and resources. Our extensive simulation based on a live video streaming service, demonstrates significant improvements in both network delay and service time. Additionally, we compare edge computing technology with the state-of-the-art cloud computing and content delivery network solutions within the context of latency-sensitive and data-intensive applications. Our results show that edge computing enhanced with suggested scheduling algorithm is a viable solution for achieving high quality of service and responsivity in deploying such applications.

1 INTRODUCTION

The enhancement of smart devices and network technologies has enabled new internet-based services with strict end-to-end latency requirements (Bilal and Erbad, 2017), for example, 360 degrees video and online gaming. Edge Computing is a promising solution for the provision of latency-sensitive applications since it offers a set of enabling technologies that moves both computation and storage capabilities (i.e., micro cloud data centers) to the edge of the network (Shi et al., 2016). This change allows deployment of services close to end users, which reduces their response time.

To effectively exploit the advantages of Edge infrastructure, service instances should be scheduled on the node which better fits service requirements, taking into account contextual knowledge, i.e., user, application and network information (Wang et al., 2017). Indeed, defining optimal placement not only allows to maximize user experience by reducing end-to-end latency, but it also optimizes network traffic by reducing bandwidth consumption and related costs, and improves energy efficiency by reducing the need to offload tasks to the cloud which incurs higher energy consumption compared to micro data centers.

Currently, few existing scheduling solutions for Edge Computing are related to Internet-of-Things, mostly focusing on the response time between service components rather than the optimization of user-related metrics (end-to-end service response time) (Skarlat et al., 2017). Scheduling approaches devised for similar paradigms, in particular cloud and content delivery networks (CDNs) are characterized by infrastructure limitations which make them unfit to meet the requirements of this type of latency-sensitive services. Specifically, cloud solutions are characterised by large distances between cloud data centers and final users, incurring high network delay that considerably degrades the service quality experienced by end users. Moreover, due to high amount of traffic generated by this kind of services, there is a considerable risk of network congestion as well as a bandwidth waste (Wang et al., 2017). In the case of CDNs, any computational tasks must be carried out in the cloud, and a distributed network of cache servers, which is located in close proximity to users, is only used for disseminating output data (e.g. web objects, rich media, etc.). However, servers are still several hops away from final users and they need to process requests on cloud servers and therefore do not considerably reduce the network delay when offloaded task is computationally heavy (Bilal and Erbad, 2017).

In this paper, we propose a novel score-based

Scoca, V., Aral, A., Brandic, I., De Nicola, R. and Uriarte, R.
Scheduling Latency-Sensitive Applications in Edge Computing.
DOI: 10.5220/0006706201580168
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 158-168
ISBN: 978-989-758-295-0

edge scheduling framework specifically designed for latency-sensitive services in Edge. The proposed technique is latency, bandwidth, and resource aware. It schedules each service instance on the VM type whose computing and network capabilities can optimize service response time experienced by end users. First, eligibility of available VM types on edge nodes are evaluated for hosting given service based on VM network and resource capabilities, and then such services are scheduled in the most suitable VMs according to eligibility scores, guaranteeing optimal service quality. We validate our approach based on a live video streaming service and evaluate how different deployment solutions, namely Cloud, CDN and Edge, affect user response time in an latency-sensitive and data-intensive scenario. The obtained results show that edge-based solutions effectively improve user response time, highlighting the need of a suitable scheduling algorithm to obtain optimized results. Moreover, we compare the performance of our approach with (Duong et al., 2012), a well-known latency-sensitive scheduling algorithm for clouds, which aims to minimize the overall delay experienced by end users by reducing the users' request waiting time, without explicitly considering network latency. The results show that edge scheduling algorithms must take also network latency into account to minimize the overall service delay experienced by end users.

Overall, the main contributions of this paper are: (i) a novel Edge scheduling framework for latency-sensitive services; (ii) the performance evaluation of different deployment solutions, namely cloud, CDN and Edge, for latency-sensitive services, which shows the advantages of Edge computing in this context; and (iii) the evaluation of the proposed framework in comparison to a solution for latency-sensitive services in clouds, which shows the benefits of using an algorithm specifically devised for this architecture.

The rest of this paper is organised a follows. In the next section, we present the related works; in Sec.3 we provide a motivating example discussing the benefit of edge-based deployment approach for latency sensitive applications such as a live video streaming service; in Sec.4 we present the scheduling approach we devised; in Sec.5 we describe the experimental setup used for the evaluation of our approach together with the results we obtained, whereas conclusions and future works are reported in Sec.6.

2 RELATED WORKS

Effectiveness of edge technologies has already been proved in many use cases, as reported by Wang et al. in (Wang et al., 2017), however there are still open research challenges to be tackled such as service scheduling on edge nodes. Recent works such as (Zhao et al., 2015; Guo et al., 2016; Mao et al., 2016) faces edge computation offloading problem, that is the decision of scheduling a task on mobile device or local/internet cloud. Nevertheless, these works do not take into account the actual service scheduling between edge nodes.

In the area of Fog Computing, in (Skarlat et al., 2017), authors defined a scheduling approach for the placement of service modules on fog nodes, focusing only on the optimization of response time among components, without taking into account the end-to-end service time. Aazam et al. in (Aazam and Huh, 2015), instead, defined a resource estimation and pricing model for IoT, that estimates the amount of resources to be allocated for a given service, again without providing an solution for the selection of a node where to allocate the service.

Scheduling of latency sensitivity services, instead, has been widely studied in Cloud. For example, VM placement solutions for distributed clouds have been developed in (Papagianni et al., 2013; Aral and Ovatman, 2016). These approaches seek for optimal VM placement on physical nodes which minimizes the network latency among them. However, these mapping methodologies rely only on service requirements and physical infrastructure knowledge without considering user-related information, such as geolocalization, which is a critical feature for the scheduling process in edge. Hence, they are not directly adaptable to our problem. In the context of single cloud providers, authors in (Piao and Yan, 2010; Zeng et al., 2014) developed service component placement methodologies that, similarly to the distributed scenario described before, optimize the service placement among servers in a single data center minimizing the transfer time between service components. Therefore, also these approaches do not take into account users information which would make their application suitable for service scheduling in edge.

Authors in (Duong et al., 2012), however, integrated user information in the service allocation methodology they proposed. Indeed, they defined a provisioning algorithm based on queuing theory to identify the number of VMs to be deployed in order to minimize the user waiting time. However, this approach, intended for IaaS clouds, only define the number of VMs needed to cope with the incoming

load and is still missing VM placement policies which would lead to sub-optimal results in the case of edge computing.

3 MOTIVATING EXAMPLE

Live video streaming services, allow to stream a video recorded by any device equipped with camera in nearly real-time to a large number of mobile or desktop audience globally. Currently, due to the advances in networking technologies these services are becoming very popular and attracting the attention of big companies such as Twitter, Facebook and Google who developed their own live video streaming services. To better understand what are the challenges underlying this kind of services, let us first quickly describe the live streaming service workflow depicted in Fig. 1.

The first step is the encoding of input video in a high quality stream, using either local camera encoder or an external one installed in remote server. Afterwards, the encoded video is given as an input to the transcoding operations which create streams in different resolutions and bitrates on the fly. This process is fundamental in order to reach a very broad audience. Indeed, creating new streams at various bitrates and resolution allows an adaptive distribution of the video content according to the device used to access the stream and bandwidth condition, guaranteeing a high quality of experience for many users. The multiple streams in output from the transcoding process are then processed by a media server responsible of packaging the video streams, according to the specifications (e.g., chunk length, codecs and container) defined by the streaming protocols (e.g., HLS, HDS, etc.) supported. The video chunks are then ready to be directly delivered to end users according to their location, bandwidth and streaming protocol implemented by the media player they use, or first disseminated on multiple servers closer to them.

Although the service workflow may look quite simple, all the steps described are characterized by some critical issues that will affect the overall user engagement to the video stream. Factors impacting user engagement, as reported in (Dobrian et al., 2011), are the join time (i.e., the duration from the player initiates a connection to a video server till the time the play starts), the buffering ratio (i.e., percentage of the total streaming session time spent in buffering) and the video quality. All these metrics are directly affected by the transcoding and distribution steps of the streaming workflow, which strictly rely on the service deployment solution adopted. Currently, streaming

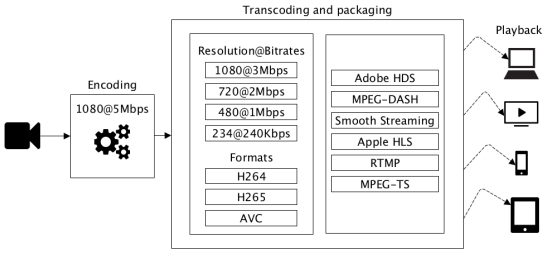


Figure 1: Live video streaming workflow.

service providers adopt cloud based deployment solutions in order to face the high and quickly changing processing resource and bandwidth requirements. In the simplest cloud based architecture, as depicted in Fig.3, the incoming video stream is usually encoded in a high quality video locally and then uploaded to a cloud server for transcoding and packaging operations.

In this way, exploiting the elasticity provided by the cloud infrastructure, it is possible to dynamically manage the resource needs according to the current service load. However, this deployment solution is not the best approach to maintain a low buffering ratio and a high video quality. Whereas extensive computation required for the transcoding operations can be managed according to the current needs, the best effort delivery of the packets from a cloud server to final users can still incur downgraded video experience. Indeed, the path between users and the stream origin server in the cloud usually involves multiple hops and then the probability of experiencing link congestion along the route is high. Congested links incur a low throughput and high packet loss and delay jitter, introducing high network delay and discontinuous packet delivery rate, causing a drop in the video quality.

To face these network issues, the most widely adopted solution currently is content delivery networks (CDNs) for the distribution of streaming contents. In this scenario, as depicted in Fig.4, all the transcoding and packaging operations are still executed on the cloud, but the media contents are then distributed over a network of cache servers located closer to end users. In this way, user requests are automatically directed to the closest cache node, which forward to the cloud only the requests whose content is not already available on that node. Therefore, an improved bandwidth consumption is provided and then the risk of network congestion is lower, reducing the buffering ratio while increasing the video quality experienced.

Nevertheless, CDNs have originally been designed for the distribution of static contents and adopting them for the distribution of dynamic contents as live video streams is not straightforward. The

main issues arise from the volume and requirements of live video streaming users. Indeed video content accounts for 70% of whole Internet traffic (Bilal and Erbad, 2017), which is much higher compared to the traffic generated by other web-applications. Moreover users demand high quality video, instant start up times and low buffering ratio, requirements which bring to high demand of bandwidth, whereas CDN aims to minimize costs and they may not meet these high-demanding requirements. Finally, a CDN-based distribution can easily become too costly as the number of streams and viewers increases, due to the high bandwidth costs.

To face the issues of the approaches described above, the set of innovative technologies introduced by the Edge paradigm may provide the solution needed. Indeed, edge solutions can be used to tackle various challenges currently faced by media and video streaming applications in order to improve the service quality experienced by end users. A possible edge-based video streaming platform is depicted in Fig.2.

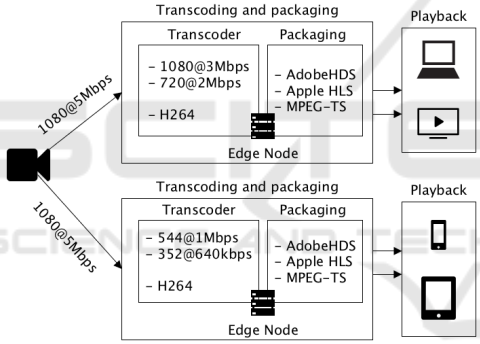


Figure 2: Edge-based platform for live video streaming services.

With edge computing, incoming video is first encoded in high quality, either with the camera encoder or with an encoder installed in the closest edge node, and then distributed over a set of edge nodes responsible for both video transcoding and content distribution. In this way, video will be encoded on the fly and delivered to the viewer, removing all the delay introduced by fetching video content from central cloud. We also assume that there are different transcoder configurations related to different device type (e.g., smartphone, tablet, IPTV, etc.) that will transcode the input video in multiple streams suitable for a given specific device. Therefore, even though edge nodes cannot provide the same computing power of cloud nodes, distribution of individual encoders, each for a specific type of device, makes computational requirements of transcoding operation suitable for edge nodes.

Overall, the main advantages of this approach are the reduction of end-to-end latency and bandwidth consumption. Indeed, strict proximity of end users to the nodes delivering video content reduces delay due to both a shorter physical distance and a reduced probability to face network congestion, since the number of links and hops along the route is smaller compared to the cloud- and CDN-based approaches. Moreover, bringing transcoding process to the edge further contribute to lower delay since video is processed and packaged on that node without the need of fetching missing content from an origin server in a cloud data center.

4 A SCHEDULING FRAMEWORK FOR EDGE COMPUTING

In this section, we describe the main components of a novel scheduling framework, depicted in Fig.5, for latency-sensitive services in Edge computing.

4.1 System Overview

A cloud provider extends the cloud infrastructure with a set of edge nodes deployed at the edge of the network. These nodes are geographically distributed according to the model described in (Hu et al., 2015), in proximity of multi Radio Access Technology (RAT) base stations (BSs), which provide access to the core network to both user equipment and edge nodes. We assume that each node is a micro data center that, leveraging on virtualization technologies, grants access to its resources by means of virtual machines (VMs). Therefore, each node defines a virtualized environment that allows the deployment of a service instance within an individual VM. Since different services have different computing requirements, in this scenario we assume that a node can provide different type of VMs. To provide edge services, the application service provider (ASP) requests to a provider the instantiation of a service in a VM compatible with the service requirements. Since in our work we consider

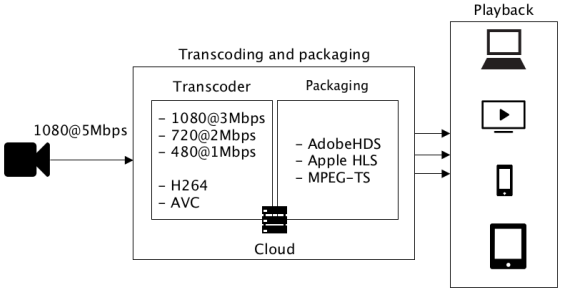


Figure 3: Cloud based solution for live streaming services.

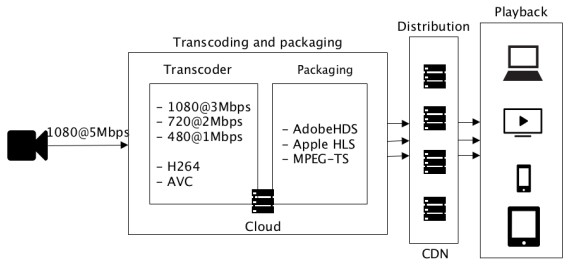


Figure 4: Delivery networks for streaming media contents.

only latency-sensitive services, the main service requirement the provider has to guarantee is the service response time, that is, the time elapsed from a user sending a request and receiving a reply. In this scenario, the network delay and the processing time are the main factors that affect this metric. The former refers to time necessary to a user request from the user device to reach the edge node of the service and it is determined by user-node physical distance, queuing and processing delay of each hop in the network route and the route's available bandwidth. The request processing time is strictly related to the VM and service specifications and refers to the time to elaborate a request. In this context with different VM and service specifications, further analysis of computing performance is needed to select a VM type that can optimize this metric.

Our goal is to design a scheduling framework that takes into account the edge computing characteristics to maximize the service quality experienced by end users of a latency-sensitive application. To that end, we defined a scheduling framework, described in Alg.1, that first evaluates both network and computational capabilities of edge nodes for each incoming service s, by calculating a quality score denoting the eligibility of that VM type to host s; and then schedules the services while maximizing the total overall quality score of the chosen VMs optimizing the service quality for the end users.

4.2 VM Evaluation

4.2.1 Low-latency Path Detection

Predicting network delay between users and edge nodes provides useful insights to understand if a given node can meet the network delay requirements of the service to be scheduled, and at runtime to select the communication routes that minimize the delay. Since several monitoring techniques are available for the network latency estimation, for example, by sending probes between nodes and then measuring the latency (i.e., active monitoring) or capturing information about network paths directly from the network devices (i.e., passive monitoring) (Yu et al., 2015), we assume that network delays between edge nodes can be measured and we use them in our scheduling approach.

We model our edge network as a weighted graph were each node denotes a edge data center in the network, the link between two nodes represents the actual connection between data centers and the link weight describes the estimated network latency between them. We assume, then, that each user connects to the closest BS and we group together users connecting to the same BS (line 2). Therefore, for each user group g we detect the lowest latency path from g to a every node n applying the Dijkstra algorithm on the graph modelling our network, considering as source node the node representing the edge data center co-located with the BS of the group (line 5). This set of low latency paths will be used for the VM evaluation process described in the next section.

4.2.2 Network and Resource Evaluation

To identify the most suitable VM to a given service, we defined an evaluation process which computes, for each type of VM, a quality score q_v combining connectivity $q_{n,l}$, bandwidth $q_{n,bw}$ and resource $q_{v,res}$ scores.

The *connectivity score* $q_{n,l} \in [0,1]$ assesses the quality of the connectivity of a VM v by evaluating the quality of the network routes connecting user groups to the node n running v. The input data for this process is the set of low-latency paths computed in Sec. 4.2.1. Therefore, for each network path connecting a given user group g to the n, we evaluate the delay according to the network delay requirements by computing a quality score $q_{n,l,g} \in [0,1]$ using a utility function previously defined by the provider (line 7). The output of this path-based evaluation process is a set of quality scores $\{q_{n,l,g_1}, q_{n,l,g_2}, \ldots, q_{n,l,g_n}\}$, which will be used to compute the connectivity final score $q_{l,n}$ as the mean value of these scores weighted by the number of users belonging to each group (line 9).

The *bandwidth score* $q_{n,bw} \in [0,1]$ assesses the available bandwidth of the node n running the given VM v. The available bandwidth on paths connecting the users to n represents one of the main factors affecting the overall service quality, as already motivated in Sec.3, therefore information about the node capacity can improve the service scheduling. We compute a per path *bandwidth score* $q_{n,bw,g} \in [0,1]$, assessing the quality of the bandwidth of each low latency route in input (line 8). Similarly to the latency evaluation process, we compute the final bandwidth quality score $q_{n,bw}$ as the average of the single path quality scores q_{n,bw,g_i} weighted by the number of users in each group

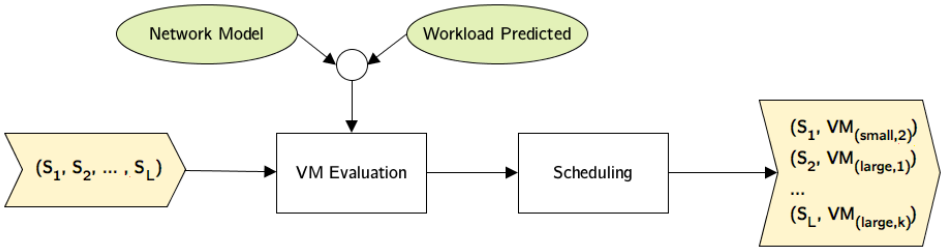


Figure 5: Scheduling framework.

(line 10).

The *VM resource evaluation* is carried out by measuring the service load that a given VM type v can handle and the expected overall service load. Indeed, for latency sensitive applications, underestimating the computational resource necessary for executing the service may increase considerably the service processing time and the overall end-to-end response time. Our approach, then, evaluates the computational resources of a VM v computing a score $q_{v,res} \in [0,1]$ by means of a utility function defined by the provider, comparing the number of user requests that v can handle $\tilde{w}$ with the overall number of requests expected w (line 12).

Finally, the overall *quality score* of a given VM type $q_v \in [0,1]$ is calculated as the harmonic mean of connectivity, bandwidth and resource quality scores (line 13). Despite the fact that the harmonic mean behaves as the arithmetic mean by giving equal weight to both scores when they are similar, it favours the smaller value when the gap between them increases. This ensures that VMs with a very high and a very low score are penalized, which emphasises the need of sufficient network and computational resources. The evaluation output is the set $Q = \{q_{v_1}, \ldots, q_{v_k}\}$ of VM type quality scores that will be used by the scheduler for service deployment (line 14).

4.3 Scheduling Approach

Given the set $s \in \mathcal{S}$ of services, the provider schedules each service instance on the VM type $v \in \mathcal{V}$ which can guarantee an enhanced end user service quality. We compute for each v a quality score $q_{s,v}$ using the approach described in Sec.4.2.2, which is based on the VM computing specifications and the network capabilities of the node hosting that VM. Then, the service instances are scheduled to maximize the overall quality of the selected VMs, guaranteeing an enhanced service quality to end users.

We model the optimisation problem as a binary integer linear programming problem as reported in formulation below. Binary variables $x_{s,v}$ model the placement of a service s on the VM type v, and assume value 1 only when the service is actually scheduled on that VM. The coefficients are the VM quality scores computed by the VM evaluation process previously described, which measures the suitability of the VM type v in hosting the service s. The cost function (I) aims to maximize the quality of the final scheduling, assigning at each service s the most suitable VM.

$$\begin{aligned}
&\underset{\bar{x}}{\text{maximize}} && \sum_{s\in\mathcal{S}}\sum_{v\in\mathcal{V}} q_{s,v}x_{s,v} && && \text{(I)}\\
&\text{s. to} && \sum_{v\in\mathcal{V}} x_{s,v} = 1 && \forall s && \text{(II)}\\
& && \sum_{a\in\mathcal{S}} x_{s,v} = k_v && \forall v \in \mathcal{V}_n && \text{(III)}
\end{aligned}$$

where

$$x_{s,v} = \begin{cases} 1 & \text{if } s \text{ is scheduled on } v \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, we assume that each service has to be scheduled on a single VM, as expressed in (II), and that the number of times a VM type is assigned to new services cannot exceed the number k_v of available VM instances, defined in (III).

5 VALIDATION OF OUR APPROACH

In this section we present the experiments we carried out to analyse the effectiveness of the proposed framework in terms of service quality experienced by the end users.

5.1 Experimental Setup

The experiments evaluate how different deployment solutions affect the live video streaming service time experienced by end users and analyse the impact on its components, namely the network delay and the request processing time, with different number of users joining the video stream in input. A comprehensive description of the different scenarios considered in

Algorithm 1: Latency-Sensitive Scheduling.

Data: Service s to be scheduled
Data: Latency constraint t requested by the service provider
Data: Users' locations $U = \{u_1, u_2, \ldots, u_m\}$
Data: Nodes $N = \{n_1, n_2, \ldots, n_k\}$
Data: Nodes' location $L = \{l_1, l_2, \ldots, l_k\}$
Data: VM types $VM = \{v_{i,1}, v_{i,2}, \ldots, v_{i,q}, \forall i \in N\}$
Data: Array Q of quality scores for each VM $v \in VM$
Result: The vm_s scheduled for s.

begin
  Define a set of users' groups $G = \{g_i, |u_j - l_i| < \varepsilon \quad \forall u_j \in U, \forall l_i \in L\}$;
  forall $n \in N$ **do**
    forall $g \in G$ **do**
      Estimate the network path $p_{g,n}$ with the lowest latency $\tilde{t}_{g,n}$ between g and n;
      Estimate the available bandwidth $\tilde{bw}_{g,n}$ between users in g and the node n and the required bandwidth bw_g on $p_{g,n}$;
      Compute the latency score $q_{n,l,g} = \text{LatencyScore}(t, \tilde{t}_{g,n})$;
      Compute the bandwidth score $q_{n,bw,g} = \text{BandwidthScore}(bw_g, \tilde{bw}_{g,n})$;
    $q_{n,l} = \frac{\sum_{i=1}^{n} |g_i| q_{n,l,g_i}}{\sum_{i=1}^{n} |g_i|}$;
    $q_{n,bw} = \frac{\sum_{i=1}^{n} |g_i| q_{n,bw,g_i}}{\sum_{i=1}^{n} |g_i|}$;
    forall $v \in n$ **do**
      $q_{v,res} = \text{ComputingScore}(w, \tilde{w}_v)$;
      Compute the q_v quality score $\mathsf{Q}[q_v] = \frac{3}{\frac{1}{q_{v,res}} + \frac{1}{q_{n,l}} + \frac{1}{q_{n,bw}}}$
  $vm_s = Scheduler(\mathsf{Q})$;
  return vm_s;

our experiments is provided in Sec.5.2. We simulated these scenarios using the EdgeCloudSim simulator (Sonmez et al., 2017). This simulator extends the CloudSim toolkit (Calheiros et al., 2011) providing extra functionalities granting the modelling of networking resources and edge nodes, allowing a accurate simulation of real edge infrastructures. The results obtained, not only show the benefit of the edge platform in terms service time, but denote the need of a edge-specific scheduling solution to obtain optimal results.

5.2 Scenarios

We defined a set of simulation scenarios where we analysed the impact of different deployment solutions and scheduling policies on the service quality experienced by end users, in the context of live video streaming services. In the remainder of this section we provide a detailed presentation of the different network designs and scheduling approaches in each scenario devised.

Cloud. We consider a centralized deployment solution where both video processing and content distribution processes are carried out on a cloud data center. Essentially, given an incoming video to be streamed, all the encoding/transcoding operations are executed on a cloud data center. User requests to access the live stream are also directly forwarded to the cloud server responsible of stream distribution.

We modelled cloud resources as set of infinite VM instances whose specifications are taken from the Amazon m2.4xlarge and are reported in Tab. 1. The users-data center connection has been defined as a single link, whose capacity has been fixed at 1000 Mbps, representing the aggregation of single links connecting user access BSs to the cloud. We defined also a communication delay $\delta_{u,c} = 0.09$, which models the delay related to the queuing and processing operations and the physical distance characterizing the network route between the user u and the cloud data center c. This value has been estimated by averaging latency of ICMP requests between hosts in Europe and the Amazon Web Service instances in the same time zone.

Content Delivery Network (CDN). We defined a two-tier network architecture, characterized by an origin server in a massive cloud data-center and 10 geographically distributed replica servers. In this scenario the encoding/transcoding operations are executed in the cloud servers, while the content is distributed among the replica nodes. Users are randomly distributed in proximity of all replica nodes and the content distribution follows the schema designed in (Pathan and Buyya, 2007). Therefore, a user request is first redirected to the closest replica server and then, if the content is already cached there, it is directly returned. Otherwise, the request is forwarded to the cloud server to fetch the content.

In this scenario origin and replica servers have different purposes and then different hardware configurations. We assumed that CDN (replica) nodes are small data centers located only in strategic points,

Algorithm 2: FIXED provisioning algorithm.

Data: Set of QoS requirements offered by the ASP: $S = \{QoS(x), x \geq 0\}$
Data: Estimated mean inter-arrival times: $\frac{1}{\lambda}$
Data: Estimated mean requests durations: $\frac{1}{\mu}$
Result: Number of VMs to acquire: V

begin
  calculate the smallest number of V such that $\frac{\lambda}{V\mu} \leq 1$;
  $t = \min(QoS(x)), QoS(x) \in S$;
  $P(w(r_i) > t) = 1$;
  $\rho = \frac{\lambda}{V\mu}$;
  while $P(w(r_i) > t) > p$ **do**
    calculate $\Pi_W = \frac{(V\rho)^V}{V!}((1-\rho)\sum_{n=0}^{V-1}\frac{(V\rho)^n}{n!} + \frac{(V\rho)^V}{V!})^{(-1)}$;
    calculate $P(w(r_i) > t) = \Pi_W e^{(-V\mu(1-\rho))t}$;
    if $P(w(r_i) > t) > p$ **then**
      $V = V + 1$
  return V

such as ISP point of presences (PoPs) or at internet exchange points (IXPs). CDN servers' access bandwidths are distributed as a Pareto distribution with mean value $\mu = 500$ and each connection user-server is characterized by communication delay $\delta_{v,s} = 0.013$s. Similar to the cloud scenario, $\delta_{v,s}$ measures the expected delay due to the physical characteristics (e.g., number of hops and distance) of the route between the host and the closest CDN server. We modelled the connection between CDN servers and the cloud origin server as a high capacity link with an average bandwidth of $\mu = 750$Mbps, since we assumed that they are directly connected to the ISP backbone. We also defined a link communication delay as $\delta_{e,c} = 0.03$s, modelling the delay along the path from the CDN and the origin server based on the number of hops and the physical distance between them. Moreover, since the main purpose of CDN networks is to deliver content, the VM instances used in this scenario are storage optimized. Therefore we used the Amazon i3.large specifications, as reported in Tab.1.

Edge. In this scenario, according to the edge-based deployment solution described in Sec.3 the service is entirely deployed on the edge nodes. Therefore, both encoding/transcoding and distribution operations are executed on the edge nodes, whereas the cloud has only management functionalities.

The designed network infrastructure is composed of 20 edge nodes, each co-locate with a BS. The computing power of each node is rather limited. Each node provides 2 types of VMs, namely the Amazon m1.large and m1.xlarge instance types, but, due to limited physical resources, only 10 instances that can actually instantiated on each node. The access bandwidth of each edge node has been modelled instead as a Pareto distribution with average value $\mu =$ 375Mbps. In this scenario we also assume that the distance between edge nodes is small, in the order of 20km, allowing the deployment of high speed inter-node connections through either dedicated links or single-hop connections. Thus, we modelled the inter-node bandwidth whose capacity is distributed also as a Pareto distribution with mean value $\mu = 400$Mbps. The communication delay between the nodes i and j has been modelled, instead, by mean of a uniform distribution $\mathcal{U}[0.006, 0.009]$.

In this scenario, we assume that users can access the stream video with 3 different type of devices, namely smartphone, laptop and tablet. Therefore, according to the edge-based service deployment described in Sec.3, we assume that 3 different and lightweight streaming engines (i.e., packages responsible of encoding/transcoding operations and content distribution) have to be deployed, each one for a specific device. Each instance is responsible for transcoding the video input into multiple bit rates and resolutions suitable for a given device. We used two different approaches for the scheduling of the instances, that is Cloud-based and Edge-based, described below.

In the Edge-based scenario, we schedule the services using the approach presented in Sec. 4.3. For the VM evaluation process, described in Sec.4.2.2, we defined three utility functions as in Eq.1, Eq.2 and Eq.3 for the evaluation of the network delay, the available bandwidth and VM resource respectively.

$$u_{v,\delta}(\delta_{g,v}, \tilde{\delta}) = S\left(1 - \frac{\delta_{g,v}}{\tilde{\delta}}\right) \quad (1)$$

$$u_{v,B}(B_{g,v}, \tilde{B}) = S\left(\frac{B_{g,v}}{\tilde{B}} - 1\right) \quad (2)$$

$$u_{v,RPS}(RPS_v, \tilde{W}) = S\left(\frac{RPS_v}{\tilde{W}} - 1\right) \quad (3)$$

For the evaluation of the network route delay between a user group g and a VM v, we compute a utility score using a sigmoid function S, that takes in input the route delay $\delta_{g,v}$ and the delay requirement $\tilde{\delta} = 50ms$. Therefore, the utility function in Eq. 1 evaluates the margin between the actual delay $\delta_{g,v}$ and the requirements $\tilde{\delta}$ returning a score as close to 1 as the current delay is significantly lower than the requirements. Otherwise, low values close to 0 are

returned. Bandwidth and resource evaluation follow the same approach defined for the delay evaluation. Essentially, given the predicted available bandwidth $B_{g,v}$ on the network route from g to v the utility function returns a value as close to 1 as the value of $B_{g,v}$ is higher then the needed bandwidth $\tilde{B}$ due to the number of users in g. The resource evaluation is achieved, instead, by comparing the computing capability of v expressed in terms of request per second *RPS* with the expected number of requests $\tilde{W}$ generated by the expected workload. Finally, the scheduling is defined by the scheduling approach in Sec.4.3.

In the Cloud-based approach, instead, we adopted a cloud scheduling approach (Duong et al., 2012) that estimates the number of streaming engines instances needed to minimize the user waiting time. This approach aims to minimize the processing time deploying a number of instances based on the expected workload. The Alg. 2 shows the pseudo-code of their approach. They define a M/M/V[5] queuing model to determine the number of VMs to be instantiated in advance to guarantee a user waiting time compliant with the requirements. Three inputs are required: (i) the estimated mean inter-arrival time of requests $1/\lambda$, (ii) the estimated mean duration of requests $1/\mu$ and (iii) and the target probability p that a request has to wait more than its QoS requirement. The first step of the algorithm (line 2) computes the initial number of VMs V, such that the system utilization is lower than one. Then, the smallest waiting time t to be guaranteed by the service provider is determined (line 3). Afterwards, the algorithm updates the number of VMs to be instantiated until the probability that a new service request $w(r_i)$ has to wait more than t ($P(w(r_i) > t)$) is lower than the threshold defined by p. Finally, for each streaming engine type the number of replica is computed and then randomly distributed among the edge nodes.

Similarly to the CDN scenario, also in this scenario, users are randomly spread in proximity of all edge nodes. Therefore, a user request is first sent to the BS co-located to the closest edge node and then forwarded to the node containing the VM which hosts the encoder related to the type of device user adopted to access the video stream.

5.3 Experimental Results

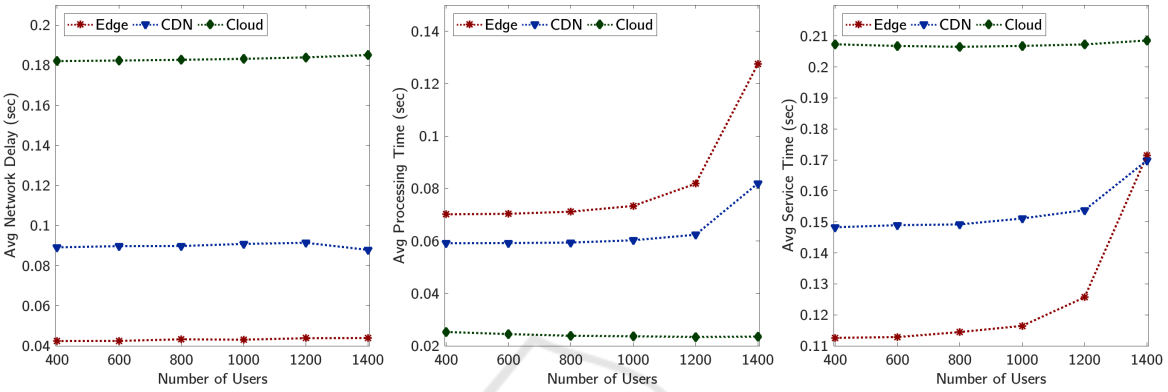
The results in Fig.6(a) show that deploying a service on the edge allows to achieve a considerable network delay reduction with respect to all the others deployment solutions. Highest reduction is around 4.5-fold with respect to cloud, and lowest is more than 2-fold with respect to CDN. Moreover, this network delay reduction is not guaranteed by only the platform itself and a suitable service scheduling algorithm is necessary as shown in Fig.7 (a). Indeed, the adoption of scheduling algorithms that do not take into account the joint information of network conditions, user requirements and workload only partially exploit the advantages introduced by the edge infrastructure, resulting in suboptimal results. Additionally, processing time represents another critical factor for the edge platform, as depicted in Fig.6 (b) and in Fig.7 (b).

The values plotted represent the time spent by the streaming engines to package the video content to be delivered. These results show how in the edge scenario we obtain the highest processing time due to the limited computational resources provided by the edge nodes. Obviously the processing time on the cloud scenario is the lowest among the three due to the high computing power characterizing cloud resources. In the CDN scenario, instead, we obtain smaller values then the edge scenario, since distributing the requests on the multiple replica server avoids the server overloading. Moreover, the limited edge node resources not only already provide the highest processing time, but they may become the system bottleneck, as the load increases. Edge results in Fig.6 (b) and Fig.7 (b) describing the processing time obtained using our approach, denote this increasing trend according to the growth of the number of input devices. Essentially, in our approach where each streaming engine instance has to process all the incoming requests from the associated user device, we experience a faster raise of the processing time. Therefore, to reduce this high processing time in the edge scenario, provider can horizontally/vertically scale the servers in the edge data centers or create new edge data centers in the overloaded locations. Results in Fig.7 (b) confirm this assumption showing that predicting in advance, the number of VM instances for each streaming engine enhances the performance, reducing the average processing time needed to elaborate each user request.

However, due to the significant network delay reduction provided by the edge-based solution, higher processing time does not affect the overall service time, if the processing capacity of the servers is not saturated, as depicted in Fig.6 (c) and in Fig.7 (c). For very high number of users, instead, higher processing time of edge introduces a system bottleneck that considerably affects the final service time experienced by end users (Fig.7 (c)). Nevertheless, results in Fig.7 (c) confirm the need for scheduling solutions that take edge specific features into account and for an infrastructure with processing capacity compatible with number of users in the edge data centers to obtain good performance in edge scenario.

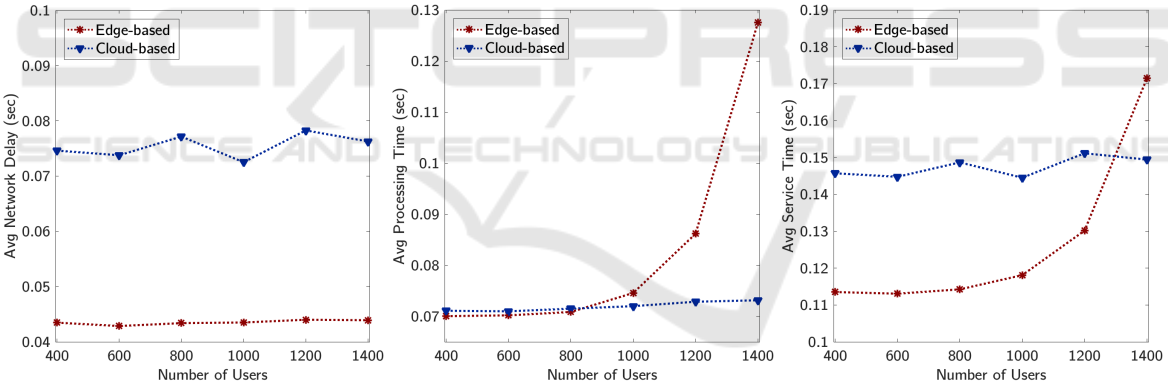
Table 1: Virtual machines specifications.

	m1.large	m1.xlarge	m2.4xlarge	i3.large
CPUs	4	8	26	2
CPU MIPS	2400	4800	20000	2400
RAM(GB)	8	16	70	15
Storage(GB)	2x420	4x420	2x840	unlimited
Price (USD/h)	0.17	0.35	0.98	0.15



(a) (b) (c)

Figure 6: Average network delay (a) average processing time (b) and average service time (c) experienced by end users in the scenarios described in Sec. 5.1.



(a) (b) (c)

Figure 7: Comparison of the average network delay (a) average processing time (b) and average service time (c) in the edge scenario using a cloud scheduler and the edge scheduler we developed.

6 CONCLUSIONS

Edge Computing is a new computing paradigm which allows the efficient deployment of latency-sensitive services in order to meet the strict requirements characterizing these services. However, an efficient service scheduling within the edge nodes is vital to fulfil the requirements. Therefore, in this work we focus on the edge service scheduling problem and propose a service-driven approach that aims to maximize the service quality experienced by end users by deploying services on the most suitable VMs in terms of computational and network resources.

We propose a two stages score-based algorithm that first evaluates the eligibility of each available VM type to host a given service, assigning to each VM type a quality score, and then schedules the services in order to maximize the total score of the chosen VMs, guaranteeing higher service quality for end users.

We evaluate the performance of different deployment solutions, namely edge, CDN, and cloud, for latency-sensitive applications. The results suggest

that edge is the best solution in this context. Then, to validate our framework, we compare the average response time, processing time and network delay experienced by the users of our approach with a related work, where our solution showed a much better performance. It is also clear that, promised benefits of edge computing can only be achieved when effective scheduling algorithms that consider its peculiar features are implemented.

We believe our work will bring more research and industry attention to this challenge and leverage the adoption of edge computing. As future work, we plan to enhance our algorithm by decentralizing the optimization, adding another decision output for the number of instances of services, and taking vertical and horizontal scaling into consideration.

ACKNOWLEDGEMENTS

This work has been supported by the Haley project (Holistic Energy Efficient Hybrid Clouds) as part of the TU Vienna Distinguished Young Scientist Award 2011, by the Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015, and by the Italian National Interuniversity Consortium for Informatics (CINI)

REFERENCES

Aazam, M. and Huh, E.-N. (2015). Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, pages 687–694. IEEE.

Aral, A. and Ovatman, T. (2016). Network-aware embedding of virtual machine clusters onto federated cloud infrastructure. *Journal of Systems and Software*, 120:89–104.

Bilal, K. and Erbad, A. (2017). Edge computing for interactive media and video streaming. In *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*, pages 68–73. IEEE.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1).

Dobrian, F., Sekar, V., Awan, A., Stoica, I., Joseph, D., Ganjam, A., Zhan, J., and Zhang, H. (2011). Understanding the impact of video quality on user engagement. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 362–373. ACM.

Duong, T. N. B., Li, X., Goh, R. S. M., Tang, X., and Cai, W. (2012). Qos-aware revenue-cost optimization for latency-sensitive services in iaas clouds. In *Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on*, pages 11–18. IEEE.

Guo, X., Singh, R., Zhao, T., and Niu, Z. (2016). An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–7. IEEE.

Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V. (2015). Mobile edge computing—a key technology towards 5g. *ETSI White Paper*, 11(11):1–16.

Mao, Y., Zhang, J., and Letaief, K. B. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605.

Papagianni, C., Leivadeas, A., Papavassiliou, S., Maglaris, V., Cervello-Pastor, C., and Monje, A. (2013). On the optimal allocation of virtual resources in cloud computing networks. *IEEE Transactions on Computers*, 62(6):1060–1071.

Pathan, A.-M. K. and Buyya, R. (2007). A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 4.

Piao, J. T. and Yan, J. (2010). A network-aware virtual machine placement and migration approach in cloud computing. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 87–92. IEEE.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

Skarlat, O., Nardelli, M., Schulte, S., and Dustdar, S. (2017). Towards qos-aware fog service placement. In *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*, pages 89–96. IEEE.

Sonmez, C., Ozgovde, A., and Ersoy, C. (2017). Edgecloudsim: An environment for performance evaluation of edge computing systems. In *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*, pages 39–44. IEEE.

Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., and Wang, W. (2017). A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, 5:6757–6779.

Yu, C., Lumezanu, C., Sharma, A., Xu, Q., Jiang, G., and Madhyastha, H. V. (2015). Software-defined latency monitoring in data center networks. In *International Conference on Passive and Active Network Measurement*, pages 360–372. Springer.

Zeng, L., Veeravalli, B., and Wei, Q. (2014). Space4time: Optimization latency-sensitive content service in cloud. *Journal of Network and Computer Applications*, 41:358–368.

Zhao, T., Zhou, S., Guo, X., Zhao, Y., and Niu, Z. (2015). A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing. In *Globecom Workshops (GC Wkshps), 2015 IEEE*, pages 1–6. IEEE.