Future Generation Computer Systems 112 (2020) 982–995

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

RL-OPRA: Reinforcement Learning for Online and Proactive Resource Allocation of crowdsourced live videos

Emna Baccour [a,*], Aiman Erbad [a], Amr Mohamed [b], Fatima Haouari [b], Mohsen Guizani [b], Mounir Hamdi [a]

[a] College of Science and Engineering, Hamad Bin Khalifa University, Qatar
[b] CSE Department, College of Engineering, Qatar University, Qatar

ARTICLE INFO

Article history:
Received 23 February 2020
Received in revised form 16 June 2020
Accepted 24 June 2020
Available online 29 June 2020

Keywords:
Live streaming
QoE
Geo-distributed clouds
Machine and reinforcement learning

ABSTRACT

With the advancement of rich media generating devices, the proliferation of live Content Providers (CP), and the availability of convenient internet access, crowdsourced live streaming services have witnessed unexpected growth. To ensure a better Quality of Experience (QoE), higher availability, and lower costs, large live streaming CPs are migrating their services to geo-distributed cloud infrastructure. However, because of the dynamics of live broadcasting and the wide geo-distribution of viewers and broadcasters, it is still challenging to satisfy all requests with reasonable resources. To overcome this challenge, we introduce in this paper a prediction driven approach that estimates the potential number of viewers near different cloud sites at the instant of broadcasting. This online and instant prediction of distributed popularity distinguishes our work from previous efforts that provision constant resources or alter their allocation as the popularity of the content changes. Based on the derived predictions, we formulate an Integer-Linear Program (ILP) to proactively and dynamically choose the right data center to allocate exact resources and serve potential viewers, while minimizing the perceived delays. As the optimization is not adequate for online serving, we propose a real-time approach based on Reinforcement Learning (RL), namely RL-OPRA, which adaptively learns to optimize the allocation and serving decisions by interacting with the network environment. Extensive simulation and comparison with the ILP have shown that our RL-based approach is able to present optimal results compared to heuristic-based approaches.

1. Introduction

Recently, crowdsourced live streaming applications have become very popular due to the technological advancement of mobile devices, high-speed communication networks, and the expansion of CPs such as YouTube and Netflix. This new paradigm has attracted the interest of millions of users [1]. As an example, Twitch.Tv stated that more than 3.7 million broadcasters used its platform in 2019 and more than 15 million viewed its videos daily [2]. Another well-known crowdsourcing application is Facebook, which attracted more than 2 billion active users in 2018 [3], out of which 78% watch live videos and 1 in every 5 broadcasted videos is live [4].

The crowdsourcing live streaming services have unique properties compared to single and multi-source broadcasting. First, the new generation of broadcasters and viewers are extremely heterogeneous in terms of device capacities and network conditions, in addition to their high and global geo-distribution. For instance, Twitch.Tv is broadcasting live contents coming from more than 100 countries with more than 150 different qualities [5]. Second, the behavior of crowdsourcers is highly dynamic over time, as broadcasters can start and end their streamings randomly, which makes it hard to estimate the resource requirements. Finally, the most critical feature of crowdsourced streaming is that a broadcaster can interact with the viewers in live, making this service very sensitive to latency. To cope with these challenges, the crowdsourced streaming service [6–9] has recently presented an overwhelming interest for the industry and academia to attract more audience and increase the CPs' revenues. Recent studies [6] revealed that two key factors are responsible for enhancing the viewers' QoE: First, the authors highlighted that viewers are likely to abandon the video, if startup delays are high and would stop revisiting the live streaming platform if they experience buffering stalls. Second, another factor to enhance the QoE is providing the requested video qualities by allocating enough resources to transcode the content. Therefore, the research question will be how to achieve the best QoE while minimizing the operational cost to allocate resources and serve viewers.

* Corresponding author.
E-mail address: ebaccourepbesaid@hbku.edu.qa (E. Baccour).

https://doi.org/10.1016/j.future.2020.06.038

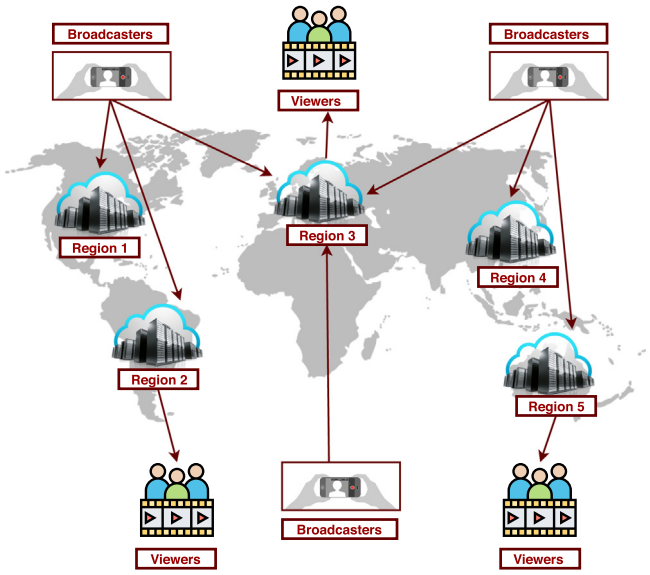


Fig. 1. Geo-distributed cloud platform for crowdsourced live streaming.

Cloud computing is a powerful technology that offers elastic and rapid resource provisioning and pay-as-you-go charging services for highly dynamic live applications, with minimum management efforts [10]. Since viewers and broadcasters are globally dispersed, an infrastructure comprising geo-distributed data centers is required to minimize the perceived delays. Accordingly, when a viewer requests a video, the CP can benefit from on-demand resource renting and redirect the request to the closest and cheapest cloud site, as seen in Fig. 1. Nowadays, crowdsourcing platforms, including Twitch.Tv, have migrated their services to geo-distributed clouds to profit from "infinite" resources and efficient charging.

To benefit from the capacity of geo-distributed cloud platforms in supporting crowdsourced services, some challenges remain to be solved. First, renting computational servers, and allocating transmission and migration resources may have different costs depending on the cloud site [11]. Using the broadcaster adjacent data center to serve viewers, may lead to increased costs. Second, compared to other streaming services, crowdsourced CP should act timely and efficiently towards dynamic situations, as the broadcasters'/viewers' behaviors are hard to predict, and live videos are time-sensitive. Thus, it is challenging to ensure a trade-off between cost and QoE by adopting static resource-provisioning approaches as done in [12]. Indeed, over-provisioning can lead to additional charges, while under-provisioning may result in lower serving efficiency. Third, the strategy of using the provisioned resources to allocate videos is very important as selecting the adequate data center affects both users and CPs. Crowdsourced live providers should avoid traditional video allocation, where each content is copied to all data centers to offer maximum QoE. Furthermore, most of the efforts, including [12] and [13], have adopted a strategy of renting servers on cloud sites, where the video is popular. Meanwhile, existing efforts [14,15] predict this popularity after receiving the feedback of viewers (e.g., reacts, joining viewers, etc.) and not at the instant of the broadcasting. Also, because of the dynamics of viewership, the popularity could change over-time, which causes allocation inefficiency and requires alteration of video location. Finally, such works opt for greedy algorithms to establish a trade-off between QoE and operational costs. These algorithms rarely achieve optimum and cannot adapt to the change of load or users' behaviors.

To summarize, it is not difficult to design an efficient allocation strategy, when the scale of users' demand or the number of broadcasted videos are fixed. The challenge is to propose an online algorithm that benefits from cloud resources to host randomly incoming live videos and handle dynamic and geo-distributed users on the fly. Meanwhile, it pursues the optimal solution presented by offline optimizations and having an overview of the system over a long period.

Compared to the previous works in the literature, we take into consideration all the discussed issues. The first step is handling the dynamic behavior of viewers. More specifically, we design a predictive model based on machine learning techniques that forecasts, the potential number of viewers in the proximity of different cloud sites, at the instant of broadcasting. To the best of our knowledge, we are the first to study geo-distributed popularity of videos, at the start of live streaming and based only on the content features (e.g., broadcaster, category, creation time and date). After assessing the viewers' distribution and the video popularity, the next step is to design an online algorithm that assists the crowdsourced live streaming system to optimize resource utilization, while ensuring the best QoE and adapting to the network dynamics and fluctuations. Recently, RL techniques have become increasingly popular, in particular for dealing with complex and large problem spaces. These techniques are able to react under unforeseen environment, by a continuous process of gaining rewards and incurring penalties for each taken action. Compared to heuristic-based approaches, RL methods do not act based on greedy decisions. Instead, they learn an optimal decision making policy from batches of environment states and knowledge of the system, which is similar to the optimization process. As a result, online and near-optimal decisions will be taken from the learned optimum policy. Another advantage of the RL approach is that it keeps learning to adapt to any system fluctuation. Our contributions in this paper can be described as follows:

- To face the challenge of geo-distribution and dynamics of online viewers, we develop a machine learning model that predicts, online and proactively, the popularity of live videos at different geo-located cloud sites, based only on the video features. This prediction is trained on a realistic and recent dataset containing streams from a well-known crowdsourcing platform: Facebook.
- We formulate the problem of crowdsourced live streaming allocation on geo-distributed cloud platform as an ILP optimization that aims to minimize the operational cost by using the predicted viewers as an input. The binary decision variables will be: choosing data centers for video allocation, deciding on video distribution between broadcaster and serving regions, and planning for sites to serve potential viewers.
- To relax the optimization problem and tackle heuristic issues, we design a novel approach based on Reinforcement Learning for Online and Proactive Resource Allocation, namely RL-OPRA. This approach learns the allocation and serving policy and takes real-time actions based on the predicted number of viewers. In this context, we define the set of states, actions, and reward function, and we use an efficient RL-approach, namely, Deep Q-Learning (DQN).
- We conduct extensive simulations to evaluate the performance of the RL-OPRA predictive model. We illustrate that the proposed approach can minimize the network cost compared to recent resource allocation systems. Additionally, we show that our popularity prediction accuracy exceeds 80%, and that the RL-approach presents close results to the optimal solution, exceeding 90%.

2. Related works

Due to the expansion of CPs such as YouTube and Netflix, the development of gaming platforms, and the emergence of multimedia technologies, such as 3D videos and Virtual Reality (VR), crowdsourcing live streaming has gained the attention of both academia and industry.

In the last decade, many existing live platforms relied on peer-to-peer (P2P) solutions [16], content distribution networks (CDNs) [17] or hybrid solutions [18]. However, with the evolution of high performance devices and the availability of internet access, video streaming services have evolved from single source broadcasting to crowdsourcing live streaming [19]. The dynamic and distributed live video sources impose critical delays and computing constraints, which draws new challenges to existing architectures to accommodate resources as close as possible to users and maximize their streaming experience. Many earlier works [20–25] derived measurements on crowdsourced live streaming platforms to understand the special features of distributed live videos, the architecture and performance of content providers, and viewers' behavior. These studies revealed, first, that live videos are broadcasted and watched from all over the world. Second, not only the number of broadcasters and viewers is enormous but it also includes heterogeneous users with different mobile and network capacities. Third, the crowdsourcers are highly dynamic since they can start and end their streaming randomly. Similarly, mobile users are frequently joining and ending their streaming sessions with a volatile online time. With the above properties, it is challenging for the CPs to meet the viewers demands with cost-effective resources and reasonable QoE.

More recently, geo-distributed cloud computing platforms have been offering flexible and elastic resource provisioning as a solution for highly distributed videos. As an example, Amazon AWS cloud [26] leases storage, transmission, and computation resources with low charges, high performance and increasing scale. Many research efforts have been conducted to orchestrate video allocation among these cloud platforms. The studies in [27] and [5] introduced cost effective geo-distributed frameworks for crowdsourced live streaming. More specifically, the work in [27] presented a generic framework that uses the adaptive leasing and price diversity to provision cloud servers and deal with service migration among geo-distributed cloud sites. Authors studied the impact of video location on the operational cost of the network and the streaming quality. Accordingly, they formulated an optimization to select the optimal number of cloud data centers for crowdsourcers to migrate their videos under lower costs. This static storage strategy cannot be applied to highly dynamic and distributed viewers, as authors focused only on the network cost and the QoE was not taken into consideration. In our work, we adopt a dynamic strategy to distribute live video streams to multiple data center sites, while considering the viewers' locations. The work in [5] covered the maximization of viewers' QoE by observing their locations and demands. Indeed, authors proposed a greedy cloud rental scheduler that allocates the popular videos in the cheapest cloud sites and offers the requested representations to the appropriate regions. Authors considered also the case where the resources are limited in different regions. However, the perceived delay is not studied as a constraint for video allocation and a metric to enhance users' satisfaction. In geo-distributed cloud platforms, redirecting requests to different data center sites is of paramount importance to serve viewers with minimum latency. In this context, authors in [28] and [29] proposed two frameworks named, CALMS and DYRECEIVE, which aim to lease cloud servers flexibly and redirect users' requests to appropriate servers in order to minimize serving delays. CALMS is proposed as a solution to accommodate viewers' location heterogeneity and mitigate the impact of requests globalization, through greedy cloud lease scheduling; while DYRECEIVE uses Lyapunov optimization framework to deal with unpredictable viewers demands/locations at a finer granularity and to adapt to the diversity of VM prices at different times and regions. The latter algorithm can have a distributed implementation. The work in [30] improved the crowdsourcing live streaming platforms by introducing the distribution of live videos among different VM instances. However, they mainly focused on the migration of the content between one fixed broadcaster and geo-distributed cloud sites. Obviously, this assumption cannot be applied to live streaming systems, where sources are distributed. To summarize, the aforementioned efforts mainly focused on offering better services to geo-distributed viewers. Still, they did not take into account the dynamics of broadcasters, which can broadly impact the performance of the crowdsourcing system. In our work, the impact of broadcaster location is taken into consideration in the migration and allocation strategy. The distribution of broadcasters is considered by the work in [31]. The authors proposed a greedy algorithm, namely GMC, that selects cloud sites to transcode live videos with the objective to minimize the operational cost and maximize the QoE. The wide distribution of viewers' locations was not considered in the above work, as will be done in our work. Furthermore, this solution opted for proactive static provisioning of resources without knowledge about distributed demands, which may incur additional costs and impose higher delays. To the best of our knowledge, only few works, including [11], were devoted to accommodate live video services in geo-distributed clouds while considering the satisfaction of both broadcasters and viewers. Particularly, authors in [11] proposed a joint optimization of data center selection and video streaming distribution, aiming at minimizing the network cost and reducing the delays to deliver live contents. However, in this work, the distribution of viewers and the popularity of videos are considered as constant from the beginning of broadcasting, which is not always the case of crowdsourced contents that gain or lose followers during the streaming. To face this problem, authors, in [32], designed a cloud assignment strategy to cope with the fluctuation of viewers' requests, where content-delivery servers are re-assigned to users based on demand change. Establishing the allocation based on initial video popularity or altering the request assignment can cost the network additional delays and extra fees to serve viewers. In our approach, we propose a prediction driven approach that estimates the potential total number of viewers near different cloud sites at the instant of broadcasting. This online and instant prediction contributes to design an efficient and proactive resource allocation.

Authors in [33] and [34] proposed prediction based approaches, that forecast and provision the required resources for future demands. Then, based on greedy heuristics, incoming videos are allocated in real-time, while being constrained by the provisioned servers. However, studying the popularity of contents is essential to use the provisioned resources more efficiently. Furthermore, greedy decisions are taken based on the current state of each cloud site. Additionally, in the literature, the cloud platform configuration is assumed to be static and predefined (e.g., number of cloud sites, costs, delays), which distinguishes our RL approach that pursues the optimal solution decisions and adapts to any environment changes through continuous online learning.

As we discussed previously, some of the aforementioned approaches exploited flexible content allocation strategies, which require knowledge of viewers behavior. Indeed, they either suppose that the number of viewers and their distribution are known from the beginning of the streaming or suppose that all viewers are located at the broadcaster neighborhood, which is not

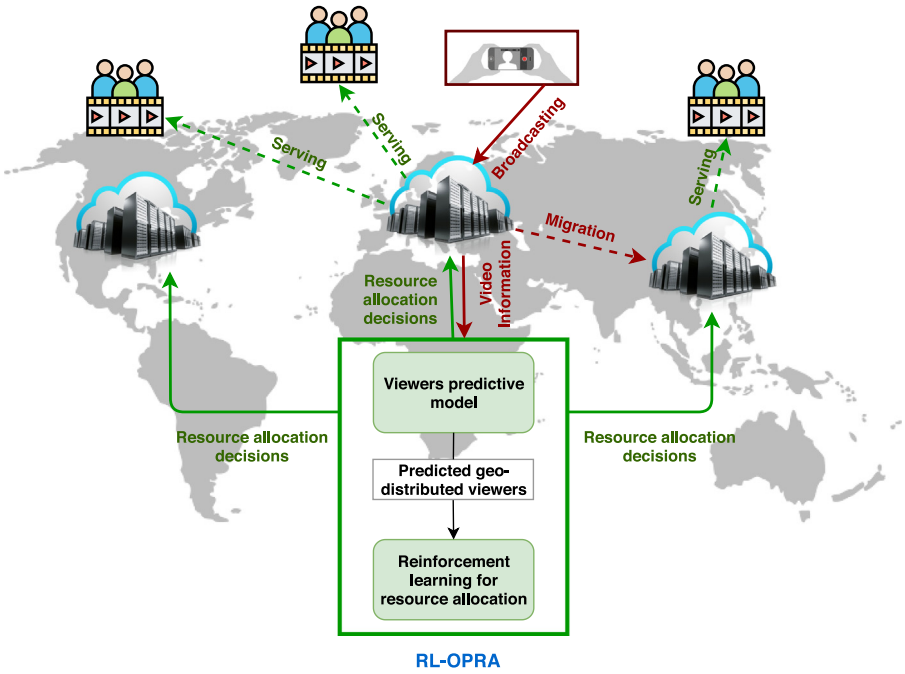


Fig. 2. System model: the RL-OPRA predictive model is deployed in a centralized master server that orchestrates resource allocation.

realistic. Such assumptions are not adequate for crowdsourcing live streaming because of the property of instant content generation and the requirement for timely actions towards dynamic situations. Therefore, an online and accurate prediction of the popularity of live videos in different cloud locations, is required.

The studies of online video popularity have been conducted in various fields such as gaming, news, TV shows and social media. Researchers in [14,15,35–37] tried to accurately estimate the popularity of a video by studying the historical parameters of the content such as the comments, shares, rating, and daily increment of view count. These works aimed at establishing an early popularity prediction. However, their methods require the observation of videos' historical features, which cannot be applied to live videos having random ending time and volatile duration insufficient to collect the required data. Authors in [38] and [39] presented a new approach to predict the popularity of contents based on visual features (video frames). Despite its efficiency, this method is complex and not always efficient for live videos. The described approaches are different from our work where the prediction is done at the instant of broadcasting based only on the metadata of the video. As claimed by the authors, the work in [40] was the first to predict viewing count for online videos. However, the study forecasted the total number of future views, which is not useful for geo-distributed resource allocation. To the best of our knowledge, we are the first to predict the number of viewers near each geo-distributed cloud site for each incoming live video, in order to timely allocate resources at the proximity of potential viewers.

In summary, the novelty and contributions of this paper are: (1) The prediction of distributed video popularity at the instant of broadcasting, based only on metadata of the stream; (2) The formulation of a strategy that minimizes the operational cost of the network and maximizes the QoE, while taking into consideration both viewers' and broadcasters' dynamics; (3) The design of an online reinforcement learning based solution, characterized by its performance compared to the optimal framework and by its continuous learning of system fluctuation.

3. QoE-aware resource allocation for crowdsourced live streaming

3.1. System model

As illustrated in Fig. 2, our crowdsourcing live streaming system is deployed in a geo-distributed cloud platform composed of multiple data centers and located in different geographical regions. These data centers cooperate to satisfy the demands of viewers and broadcasters. We suppose that our platform consists of N geo-distributed regions; in each region, one data center is set to offer services to end-users located in the same region or distributed in other regions. Several services are indispensable to run the live streaming system smoothly, namely, video computation to transcode a content from a higher quality to a lower one, video migration to distribute the content to other regions, and request response to serve viewers. We assume that these services' charges are potentially different across cloud regions. The first service is charged by time unit while the rest are charged depending on the traffic volume. In this paper, we follow the charging model, EC2 [41] and S3 [42], of Amazon, which is a leader in commercial cloud services. The goal of this work is to reduce the total operational cost, while offering the best QoE of live streaming viewers. As we discussed previously, to ensure minimum networking cost, it is required to allocate live and proactively the exact number of servers that achieves the targeted QoE. Hence, in our work, we propose a RL-OPRA predictive model, which we deploy in a centralized master server (see Fig. 2). More precisely, when the platform user broadcasts a video in real time, the content is hosted by default in the adjacent data center and its information (category, broadcaster, creation time,etc.) are reported to the master server. The RL-OPRA predictive model is composed of two phases: In the first phase, we will predict the expected number of viewers to join the streaming, related to each region. Based on the prediction results, the second phase will decide on the fly where to migrate the videos replicas and from which geo-located data center to serve viewers, in order to reduce the perceived delays and the experienced stalls with maximum cost gain. The trade-off between cost and delay will be

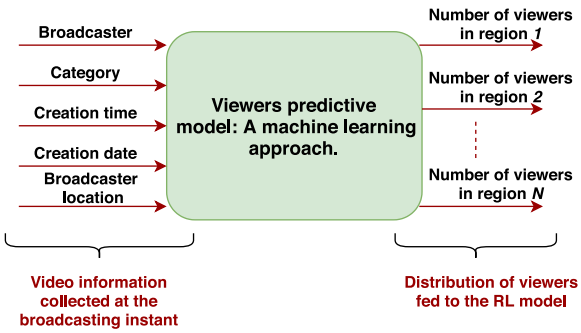


Fig. 3. Illustration of the viewers predictive model.

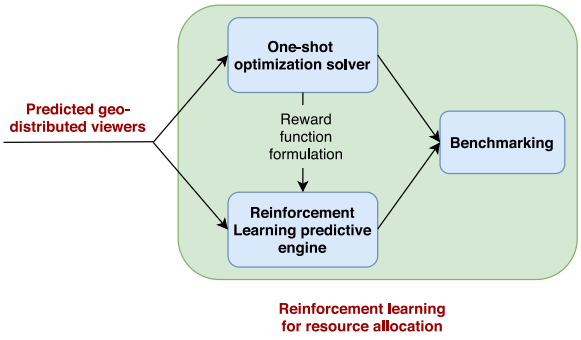


Fig. 4. Illustration of the RL predictive model.

handled by an ILP optimization and an RL framework that serves as an online solution. This framework mimics the optimization process by learning the optimal policy from batches of videos to give best allocation decisions online. Another advantage of the RL-OPRA framework is that it learns to adapt to the environment changes. For convenience, Table 1 presents some key notations used in ulterior sections.

3.2. Online prediction of geo-distributed viewers: phase 1

In this work, we adopt Facebook as a crowdsourcing live streaming platform and we use the Facebook dataset [25] recently collected by our team. This dataset contains more than 3 million live video streams collected in January, February, May, June and July 2018. Each live video stream is collected with plenty of features, including the broadcaster, the category, the description, and the number of viewers and their locations. For our first prediction phase, we select five relevant features, which are the creation time, creation date, the category, and broadcaster ID and location (see Fig. 3). We choose this list of features as it can be collected from the video at the broadcasting instant, which is not the case of the number of likes, comments, length and frames of the content used by other related works to predict the popularity.

Using these data, we predict the popularity of live contents at each region in terms of number of viewers. However, we need to pre-process the raw data. First, each viewer's location is mapped to one Amazon Web Services (AWS) cloud site [43], by finding the nearest site to him/her. In this way, for every video record, we define the number of viewers related to each cloud site. Similarly, the location of the broadcaster is mapped to the adjacent site. We design our system to be composed of $N = 10$ AWS sites, namely US West-California, US East-Virginia, US East-Ohio, South America-Sao Paulo, Europe-Paris, Europe-Frankfurt, China-Ninxgia, Asia-Singapore, Asia-Seoul, and Asia-Mumbai. As a next step, we convert high-cardinality features (video category and broadcaster id) to hashed feature vectors, using feature hashing methods. Moreover, we map the creation date into different week-days and cluster the creation time into 6-time intervals. Finally, we apply the categorical one-hot encoding to process the broadcaster location and the creation date and time.

As shown in Fig. 3, the regression model will be trained to produce 10 outputs. Each predicted output represents the potential number of viewers related to each AWS data center. The data is trained using three machine learning algorithms namely, Random Forest (RF), Decision trees (DT) and Multilayer-perceptron (MLP). Since it is not possible to predetermine the combination of hyper parameters that gives the best results, we test different models. We varied the forests number in RF algorithm, the max depth for DT, and the number of neurons and hidden layers of MLP. Then, based on the determination coefficient (R^2), we select the model giving the best accuracy. R^2 is calculated as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{V} (r_i - p_i)^2}{\sum_{i=1}^{V} (r_i - \bar{v})^2}, \tag{1}$$

where V is the total number of video records, r_i and p_i present respectively, the real and the predicted viewers joining the stream i, and $\bar{v}$ denotes the mean number of viewers among all records. We note that different models give accurate predictions, when R^2 approaches 1.

3.3. Resource allocation of crowdsourced live video streaming: phase 2

The contribution of this paper is the design of an online approach that optimally copies live video to different data center sites with different costs and different distances to viewers and proactively dispatches the potential requests incoming from predicted viewers to timely response with highest QoE and lowest charges. We, first, formulate an ILP optimization that grants the best strategy for video allocation and requests serving, while having complete knowledge about the network (video arrivals and predicted viewers), over a long period of time T. The optimization that is run offline, serves as a guideline to design the reward function of the RL-approach and a benchmark to evaluate the performance of the system (see Fig. 4).

3.3.1. Problem formulation: trade-off between QoE and cost

In the following, we will present different factors that have an impact on the users' QoE and the operational cost of services.

Service QoE: One of the characteristics that distinguishes the live streaming from other multimedia services is that the interaction delay between viewers and broadcasters (e.g. online comments and reactions) is very critical and affects the QoE greatly. Hence, we should ensure that the average delay is lower than the application required threshold to guarantee maximum satisfaction. High thresholds will be tested to see the trade-off in terms of cost. In this paper, we will consider only the round-trip delay between different cloud sites. Also, for the evaluation of QoE, we will focus only on reducing the delivery delays as much as possible and we will not consider the transcoding impact on the users' satisfaction. The main contribution of our strategy is the selection of optimal data centers to migrate and replicate contents for transcoding and determine the closer sites to viewers to serve them with minimum cost. We split the period T into equal intervals t, where $t = 0$ denotes the initial time slot. Let $V(t) = \{v_1, v_2, v_3, \ldots, v_m\}$ present the set of live videos broadcasted at the interval t. We assume that all contents have equal size Z_v and that the time interval t is enough to upload the full stream to the system. The set $Rg = \{r_1, r_2, r_3, \ldots, r_N\}$

Table 1
Table of notations.

Notation	Description
N	Number of geo-distributed cloud sites.
R^2	Determination coefficient.
V	Total number of videos.
$\bar{v}$	Mean number of viewers among all records.
T	Period of testing.
v_i	Video index.
Z_v	Size of a video.
Rg	Set of cloud regions.
r_j	Region index.
$r^b_{v_i}$	Broadcasting site of video v_i.
$r^a_{v_i}$	Allocation region of video v_i.
$r^s_{v_i}$	Viewers' region where v_i is served.
$d(r^a_{v_i}, r^s_{v_i})$	Round-trip delay between $r^a_{v_i}$ and $r^s_{v_i}$.
$\mathbb{R}$	Total rental cost.
c^j_r	Rental cost per GB in the region r_j.
$H(v_i, r^a_{v_i})$	Decision variable, indicates that v_i is hosted in the site $r^a_{v_i}$.
$\mathbb{M}$	Total migration cost.
$c^{r^b_{v_i}}_m$	Cost of migrating a copy of v_i from broadcasting site $r^b_{v_i}$.
$\mathbb{S}$	Total serving cost.
c^j_s	Cost of data transfer.
$p(v_i, r^s_k)$	Predicted number of viewers in the streaming region r^s_k.
$P(v_i)$	Set of viewers of video v_i in different regions.
$\mathbb{C}ost$	Total operational cost.
$B(v_i, r^a_j, r^s_k)$	Decision variable, indicates that v_i is served to viewers in r^s_k from the allocation site r^a_j.
$E(v_i, r^s_k)$	Binary variable, indicates that $p(v_i, r^s_k) > 0$.
$\mathbb{D}$	Delay threshold.
(S, A, R)	(State, Action, Reward).
t_s	Time-step.
IAD	Incremental Average Delay.
dr	Deferred reward.
π	RL policy function.
γ, α	Discount factor, learning rate.
Q	Action–state function.
θ	Weights of the deep network.
L	Loss function.
D	RL memory.
G	RL updating step.
Q_{target}	Update network.

denotes different cloud sites that cooperate to allocate videos, which are equal to 10 in our study. Let $r^b_{v_i}$ denote the broadcasting site, $r^a_{v_i}$ the allocation region and $r^s_{v_i}$ present viewers region. The round-trip delay, between the region $r^a_{v_j}$ where v_i is allocated and transcoded and $r^s_{v_i}$ to where the video is streamed, is denoted by $d(r^a_{v_i}, r^s_{v_i})$.

Operational cost: The CPs should rent different resources from the geo-distributed cloud platform to transcode live video streams into different qualities, dispatch it to different data centers, and distribute it to viewers to satisfy their requests. In this work, we will assume that a rented server has enough capacity to transcode one video to different requested qualities. To summarize, a CP that intends to deploy its streaming services, should pay the following operational costs: (1) rental cost, (2) dispatching and migration cost, and (3) serving cost. We assume that different charges stay constant during each studied time interval t. The operational costs are detailed as follows:

(1) Rental cost: We assume that the provisioning of resources is based on on-demand charging. This cost, namely c_r, varies depending on the cloud site and the data usage level fixed by the cloud platform. As an example, Amazon charges 0.023$ per GB for the first 50 TB, while it charges 0.021$ when exceeding 500 TB in US East Virginia region [42]. The cost/GB of renting a server for computation and transcoding tasks in different data centers r^a can be represented by:

$$\mathbb{R}(t) = \sum_{v_i \in V(t)} \sum_{r^a_j \in Rg} c^j_r(t) * Z_v * H(v_i, r^a_j), \tag{2}$$

where $H(v_i, r^a_j) = 1$ indicates that a live video stream $v_i \in V(t)$ is hosted in a server located at the cloud site r^a_j; 0 otherwise.

(2) Dispatching and migration cost: When a viewer requests a video from his/her closest data center, which is not currently serving the content, the request is either satisfied from a far-away cloud or migrated to a closer one in order to meet the overall latency requirement of the system. The migration cost is expressed as follows:

$$\mathbb{M}(t) = \sum_{v_i \in V(t)} \sum_{\substack{r^a_j \in Rg, \\ r^a_j \neq r^b_{v_i}}} c^{r^b_{v_i}}_m(t) * Z_v * H(v_i, r^a_j), \tag{3}$$

where $c^{r^b_{v_i}}_m$ is the price of migrating a copy of the content v_i from the broadcaster data center $r^b_{v_i}$ to the target site r^a_j per GB. If $r^b_{v_i} = r^a_j$, no migration is needed and $c^{r^b_{v_i}}_m(t) * H(v_i, r_j)$ should be equal to 0.

(3) Serving cost: It is called also bandwidth cost. When distributed viewers download contents from selected cloud sites, the CP should pay the cost of serving these requests. The bandwidth cost is presented as follows:

$$\mathbb{S}(t) = \sum_{v_i \in V(t)} \sum_{r^a_j \in Rg} \sum_{r^s_k \in Rg} c^j_s * Z_v * p(v_i, r^s_k) * B(v_i, r^a_j, r^s_k), \tag{4}$$

where c^j_s is the cost of data transfer from the cloud site r^a_j to the internet per GB. $p(v_i, r^s_k)$ denotes the predicted number of viewers of the video v_i, at the streaming region r^s_k. Since each stream v_i

is viewed in different geo-distributed locations, we define the set of predicted number of viewers as $P(v_i) = \{p(v_i, r_1), p(v_i, r_2), \ldots, p(v_i, r_N)\}$. $B(v_i, r_j^a, r_k^s) = 1$ indicates that a cloud site r_j^a serves the video v_i to viewers at region r_k^s; 0 otherwise.

The total operational cost for crowdsourced live streaming system deployed on geo-distributed cloud platform during the time interval t, can be calculated as follows:

$$\mathbb{C}ost(t) = \mathbb{R}(t) + \mathbb{M}(t) + \mathbb{S}(t) \tag{5}$$

The selection of cloud sites for video allocation (defined by the decision variable $H(v_i, r_j^a)$) and the redirection of viewers to be served from geo-distributed data centers (defined by the decision variable $B(v_i, r_j^a, r_k^s)$) have a large impact on perceived delays and system costs. Hence, to balance the trade-off between the network cost and the QoE, these two decision variables should be properly designed. To achieve this goal, we formulate our problem into the ILP optimization (6). In order to make reading the optimization easier, we remind that $V(t)$ denotes the total number of videos in the time interval t, Rg presents the set of cloud sites, and $r_{v_i}^a$, $r_{v_i}^b$ and $r_{v_i}^s$ denote respectively the allocation region, the broadcaster region and the streaming region of a video v_i. Furthermore, $p(v_i, r_k^s)$ denotes the number of viewers in the region r_k^s and $\mathbb{D}$ defines the delay threshold fixed by the crowdsourcing streaming platform. Finally, let $E(v_i, r_s^{v_i})$ present a binary variable, equal to 1, if the predicted viewers $p(v_i, r_{v_i}^s) > 0$ near the region $r_{v_i}^s$; and 0 otherwise.

$$\min_{H(v_i, r_j^a), B(v_i, r_j^a, r_k^s)} \quad \lim_{T \to \infty} \sum_{t \in T} \mathbb{C}ost(t), \tag{6a}$$

$$H(v_i, r_{v_i}^b) = 1 \qquad \forall v_i \in V(t), \forall r_{v_i}^b \in Rg \tag{6b}$$

$$B(v_i, r_{v_i}^a, r_{v_i}^s) \leq H(v_i, r_{v_i}^a) \quad \forall v_i \in V(t), \forall r_{v_i}^a, r_{v_i}^s \in Rg \tag{6c}$$

$$(v_i, r_{v_i}^a, r_{v_i}^s) \leq E(v_i, r_{v_i}^s) \quad \forall v_i \in V(t), \forall r_{v_i}^a, r_{v_i}^s \in Rg \tag{6d}$$

$$\sum_{r_j^a \in Rg} B(v_i, r_j^a, r_{v_i}^s) = E(v_i, r_{v_i}^s) \ \forall v_i \in V(t), \forall r_{v_i}^s \in Rg \tag{6e}$$

$$\frac{\sum_{r_j^a \in Rg} \sum_{r_k^s \in Rg} p(v_i, r_k^s) * d(r_j^a, r_k^s) * B(v_i, r_j^a, r_k^s)}{\sum_{r_k^s \in Rg} p(v_i, r_k^s)} \leq \mathbb{D} \quad \forall v_i \in V(t) \tag{6f}$$

$$H(v_i, r_j^a), B(v_i, r_j^a, r_k^s) \in \{0, 1\} \tag{6g}$$

Different constraints are described as follows: the constraint (6b) ensures that every new incoming stream is hosted by default in the adjacent data center to the broadcaster. The constraint (6c) guarantees that each video v_i is served to viewers at region $r_{v_i}^s$ from the cloud site r_j^a, only if it is allocated at this site. The constraint (6d) specifies that a video v_i is served to region $r_{v_i}^s$, only if requests are received from this region. The constraint (6e) ensures that viewers' requests can only be handled by one data center. Respecting the average delay constraint is described in constraint (6f), where the average serving time to all viewers should be lower than the required threshold. Finally, (6g) imposes that the decision variables are binary and can only take a value of 0 or 1. The problem in (6) is an NP-hard problem, which makes finding the optimal solution extremely challenging in terms of time. To reduce the complexity of the optimization, we propose an RL-based approach for online resource allocation.

3.3.2. Reinforcement learning for online and proactive resource allocation

In this section, we formulate the resource allocation of crowdsourcing live videos as a reinforcement learning process. RL can be considered as one of the machine learning paradigms, in addition to supervised and unsupervised learning. The key advantage of the RL is that it learns by interacting with its environment (geo-distributed cloud platform and crowdsourcing system) and then adapts to it. The RL system gains rewards and receives penalties for every action it takes until reaching the optimum of the policy. In this case, it becomes able to adapt to the environment, maximize the cumulative rewards on the fly, and achieve the objective. The essential elements of an RL approach can be presented as a tuple (S, A, R), which refers to State, Action, and Reward, respectively. When a state is presented to the system, an action is taken and a reward is attributed to judge the effectiveness of the prediction. This State/Action task is called a time-step or epoch. All the time-steps that occur between the initial state and the terminal state are named an episode. The goal of the RL is to maximize the total reward gained during different time-steps of an episode. Different episodes are independent and the total reward is initialized to 0, at the beginning of each episode. An intuitive idea to design the RL tuple is to define the state as the set $P(v_i)$ of predicted viewers at different cloud sites. Furthermore, the action in a time-step can be defined as the matrix of serving $B(v_i, r_j^a, r_k^s)$; introduced in the previous section. Each time-step represents an incoming video v_i. Different columns represent the data centers r_j^a that allocate and serve viewers, while the rows represent the sites r_k^s that have viewers for a video v_i. As an example, $B(v_i, r_j^a, r_k^s) = 1$ indicates that the viewers related to the site r_k^s are served by the data center r_j^a. We note that N defines the number of cloud sites deployed in our platform. The total reward will be attributed based on respecting the average delay and minimizing the cost of different actions in the episode (period t). However, this design requires the exploration of 2^{N^2} possible actions (2^{100} in case of 10 sites) to find the best allocation. In this way, the convergence to an optimal solution becomes impossible in terms of computation and time. To simplify the exploration, we redefine the tuple (S, A, R) as shown in Fig. 5 and we identify the state space, action space, and reward function as follows:

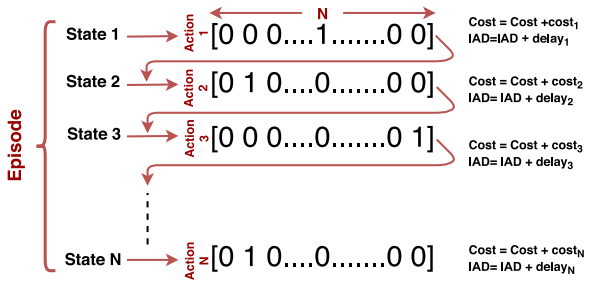


Fig. 5. Illustration of an episode.

State space: Each matrix $B(v_i, r_j^a, r_k^s)$, where v_i is fixed, will be divided into N time-steps, each row presents a separate step (see Fig. 5). The system state at the t_sth time-step ($t_s = 1, 2, \ldots, N$) consists of four components $S(t_s) = (r_k^s, p(v_i, r_k^s), \mathbb{C}ost, IAD)$, where r_k^s presents the data center index, $p(v_i, r_k^s)$ presents the number of predicted viewers at the location r_k^s, $\mathbb{C}ost$ defines the cost incurred from previous time-steps, and IAD describes the Incremental Average Delay. We update the $\mathbb{C}ost$ and IAD at each t_sth time-step according to the predicted action. The updated cost/delay will be the input to the next $(t_s + 1)$th time-step as shown in Fig. 5.

Action space: In order to reduce the operational costs and maximize the QoE, the RL agent needs to decide which cloud site will serve the $p(v_i, r_k^s)$viewers in region r_k^s at time-step t_s. An

example of an action space is represented as follows:

$$A(t_s) = [0\ \ 0\ \ 1\ \ \ldots\ \ 0\ \ 0], \tag{7}$$

where only one element of the vector should be 1, which indicates the index r_j^a of the site that will serve the viewers. If $p(v_i, r_k^s)$ is equal to 0, all elements should be null. Furthermore, if the element r_j^a is equal to 1, it means that $H(v_i, r_j^a)$ and $B(v_i, r_j^a, r_k^s)$ are equal to 1.

Reward function: When the RL agent executes an action $A(t_s)$ based on the state $S(t_s)$, a reward is attributed. A decision should be taken in each time-step while respecting three constraints, namely C1, C2 and C3, as presented in Eq. (8). C1 indicates that the sum of the action vector should be equal to 1, if there are viewers at site r_k^s, which is equivalent to constraint (6e). C2 indicates that the sum of the action vector should be equal to 0, if no request is received from the site r_k^s, which matches the constraint (6d). C3 shows that each serving decision should respect the average delay threshold, which is equivalent to the constraint (6f). Let $delay_{t_s}$ denote the incurred delay in only the current time-step t_s after serving the viewers $p(v_i, r_k^s)$. More specifically, if at a step t_s, we have $p(v_i, r_k^s)$ viewers in the region site r_k^s and the action is to serve from a cloud site r_j^a, $delay_{t_s}$ will be equal to $\frac{d(r_k^s, r_j^a) * p(v_i, r_k^s)}{\sum_{r_k^s \in Rg} p(v_i, r_k^s)}$. Next, let *IAD* denote the Incremental Average Delay of different time-steps in one episode. Particularly, *IAD* defines the average delay of the current time-step t_s and previous time-steps, meaning the sum of $\{delay_1 \ldots delay_{t_s}\}$ as shown in Fig. 5. At each t_s, we test if the incremental average delay respects the threshold. When $t_s = N$, *IAD* is equal to the total average delay to serve all viewers in all regions.

$$\begin{cases} \text{C1: if } p(v_i, r_k^s) > 0 \rightarrow \sum A(t_s) = 1, & \text{constraint (6d)} \\ \text{C2: if } p(v_i, r_k^s) = 0 \rightarrow \sum A(t_s) = 0, & \text{constraint (6e)} \\ \text{C3: } IAD + delay_{t_s} \leq \mathbb{D}, & \text{constraint (6f)} \end{cases} \tag{8}$$

We define the immediate reward in Eq. (9). If constraints C1 and C2 are not respected, the geo-distributed cloud platform will have a poor performance because requests are not satisfied or unneeded resources are allocated. Hence, we set the reward to be 0 to avoid invalid situations. If C1, C2, and C3 are respected, we attribute 1 as a reward. However, it is not compulsory that each site is served with minimum delays and it is only required that the average delay of all cloud sites respects the threshold. Therefore, if only C3 is not respected, we save the reward as a deferred recompense in a variable dr. At the end, when $t_s = N$ and the average delay *IAD* is lower than $\mathbb{D}$, dr will be added to the reward.

$$\begin{cases} \text{C1, C2 not satisfied,} & 0 \\ \text{C1, C2, C3 satisfied,} & 1 \\ \text{C1, C2 satisfied, C3 not satisfied,} & dr = dr + 1 \end{cases} \tag{9}$$

In addition to the rewards given for respecting different constraints, the geo-distributed platform charges the system for hosting copies of the live video and serving distributed viewers. Therefore, these charges are counted as penalties given to the system and added to the reward function. In this way, the RL agent tries to maximize the cumulative rewards by minimizing the penalties and selecting optimal allocations. More specifically, when $t_s = 1$, the reward $R(t_s)$ is initialized to $\mathbb{C}ost = -c_r^{r_{v_i}^b} * Z_v$, which is the cost of allocating a copy of the video by default at the adjacent data center to the broadcaster as indicated by the constraint (6b). This cost will be loaded with the state $S(ts = 1)$.

Deep Q-learning algorithm: Since our geo-distributed cloud platform is highly dynamic and the action space is highly dimensional, it is hard for the RL agent to make resource allocation decisions using traditional reinforcement learning methods. Hence, we use a Deep Q-Learning (DQN) approach [44] to determine the optimal actions and learn a policy that maximizes the cumulative rewards. The policy function is defined as $\pi : S(t_s) \times A(t_s) \rightarrow [0, 1]$, where $\pi(S(t_s), A(t_s))$ represents the probability of choosing an action $A(t_s)$ when receiving a state $S(t_s)$. In Q-leaning, the feedback for each action is represented by the action–state function $Q(S(t_s), A(t_s))$, which is expressed as follows:

$$Q(s, a) = E[\sum_{k=1}^{N} \gamma^k R(t_s + k) | S(t_s) = s, A(t_s) = a], \tag{10}$$

where $E[*]$ denotes the action expectation under the received state and following the policy function π. $\gamma \in [0, 1]$ is defined as the discount factor that serves to trade-off the immediate reward in the current time-step and the long-term reward at the end of the episode. We remind that (S, A, R) represents the state and actions spaces, and the reward function. To evaluate $Q(s, a)$, a temporal difference method is used:

$$Q(s, r) \leftarrow Q(s, r) + \alpha(R(t_s) + \gamma \max_{a'} Q(s', a') - Q(s, a)), \tag{11}$$

where $\alpha \in (0, 1]$ denotes the learning rate. In the traditional Q-learning algorithms, for each state, a $Q(s, r)$ is calculated and saved into a Q-table. Then, the action that has the larger $Q(s, r)$ is selected. However, since in our system, the state and action spaces are highly dimensional, it is challenging to save all Q-values in the Q-table. Therefore, deep learning is used for $Q(s, r)$ evaluation. We approximate $Q(s, r, \theta) \approx Q(s, r)$, where θ represents the weights of deep networks. The key idea of the DQN is to train deep networks to minimize the loss function illustrated as follows:

$$L(\theta) = E[(R(t_s) + \gamma \max_{a'} Q(s', a', \theta') - Q(s, a, \theta))^2]. \tag{12}$$

Algorithm 1 presents our DQN approach. Line 2 → line 5 illustrate the initialization phase, where the Q-network is prepared and a target Q-network is initialized with the same parameters. Then, at real time, the DQN model continues learning while taking online decisions, in order to adapt to any system change. More particularly, in line 33, the new states, actions and rewards are saved in an experience memory D. Note that the described steps to calculate the reward are summarized in line 6 → line 30. Next, a random mini-batch is sampled from the experience memory D (line 34) and target values for each tuple in the sample-batch are calculated using the target Q-network (line 37). This mechanism, known as experience replay, helps to further learn from past experience and leads to the stabilization and convergence of the learning process. While improving the main Q-network, the target network is held fixed, to stabilize the learning. Then, at each G steps, the target network is updated towards the main one, so that it always reflects the most recent knowledge (line 38) [44]. In this way, after the initialization phase, and when implementing the system in real-time, the DQN model keeps learning and adapting to any change in the environment, including costs update, demands load, and clouds number.

Algorithm 1 RL-OPRA

Initialization:
Use historical states to estimate $Q(S, A)$ values.
Save $Q(S, A)$ values and states in experience memory D.
Pre-train the DNN (Q-network) with input pairs (S, A) and the corresponding estimated $Q(S, A)$ and introduce network parameters θ.
Initialize a second target Q-network $\theta' = \theta$
DQN continuous Learning:
for each episode $v_i \in \mathrm{V}$ **do**
  $IAD = 0$, $dr = 0$
  $\mathbb{C}ost = -c_r^{r^b_{v_i}} * Z_v$
  for each time-step $t_s = 1..N$ **do**
    $delay_{t_s} = 0$
    $S(t_s) = (t_s, p(v_i, t_s), \mathbb{C}ost, IAD)$
    $A(t_s) =$arg max$Q(S(t_s), A(t_s))$
    for $k = 1 : N$ **do**
      if $t_s \neq r^b_{v_i}$ **then**
        $\mathbb{C}ost = \mathbb{C}ost - (c_r^k + c_m^{r^s_k} + c_s^k * p(v_i, t_s)) * Z_v * A(t_s)(k)$
      end if
      $delay_{t_s} = delay_{t_s} + p(v_i, t_s) * A(t_s)(k)$
    end for
    $IAD = IAD + delay_{t_s}$
    $R(t_s) = R(t_s) + \mathbb{C}ost$
    if $(\sum A(t_s) = 1$ **and** $p(v_i, t_s) > 0)$ **or** $(\sum A(t_s) = 0$ **and** $p(v_i, t_s) = 0)$ **then**
      if $IAD < \mathbb{D}$ **then**
        $R(t_s) = R(t_s) + 1$
      else
        $dr = dr + 1$
      end if
    end if
    if $IAD < \mathbb{D}$ **and** $t_s = N$ **then**
      $R(t_s) = R(t_s) + dr$
    end if
    Observe $R(t_s)$ and the next state $S(t_s + 1)$.
    Save $(S(t_s), A(t_s), R(t_s), S(t_s + 1))$ in the experience memory D.
    Sample a mini-batch of $(S(j), A(j), R(j), S(j + 1))$ from the memory D.
    Find target Q-value $Q_{target}(j)$ from target Q-network:
    $Q_{target}(j) = R(j) + \gamma max_{a'} Q(s', a', \theta')$.
    Update the target Q-network with loss function:
    $L(\theta) = [Q_{target}(j) - Q(s, a, \theta)]^2$ every G steps.
  end for
end for

4. Performance evaluation

In this section, we evaluate the performance of our machine and reinforcement learning for online resource allocation in geo-distributed cloud platform. First, we present the simulation settings, then, we discuss the simulation results.

4.1. Simulation settings

As described in Section 3.2, we used our Facebook dataset [25] to train and test our predictive models. Also, we opted for AWS as a geo-distributed cloud platform, and we chose to adopt $N = 10$ AWS cloud sites. We will show the performance of the RL-OPRA system on a period $T = 24$ h corresponding to July 3, 2018. The optimization was run on hourly-based videos' arrival; $t = 1$ h (see Section 3.3.1). The number of hourly broadcasted videos, on July 3, 2018, is presented in Fig. 6. Note that the total number of videos is equal to $V = 8015$. Furthermore, as the qualities of requested videos are out of the scope of this work, we assumed that all videos have the same size Z_v equal to 0.738 Gbit. Next, the round trip time matrix $d(r^a, r^s)$ is created by finding the ping times between different geo-distributed cloud sites [45]. Finally, we followed the charging model of Amazon [41] S3 and EC2 to calculate c_r, c_m and c_s. We considered different latency thresholds $\mathbb{D}$ to evaluate the performance of our system, specifically 60 ms, 120 ms and 220 ms. According to [11], the maximum interactive delays in live streaming systems should be around 200 ms. Higher delays will bring poor user experience. Hence, in our simulation, we chose three thresholds to show the trade-off between the perceived delays and operational costs. Regarding the RL approach, we listed the parameters in Table 2.

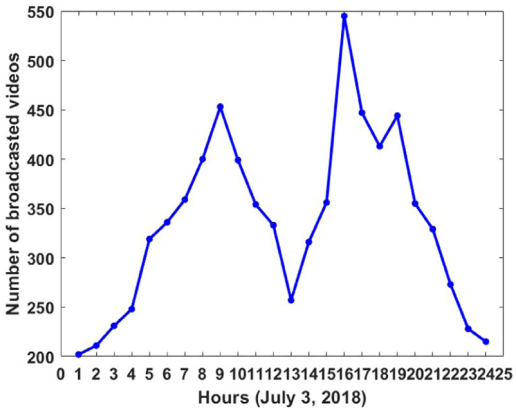


Fig. 6. Hourly broadcasted videos.

Table 2
Parameters of the RL simulation.

Parameter	Description	Value
γ	Discount factor	0.99
α	Learning rate	0.0005
M	Batch size	32
bz	Buffer size	1000
G	Network update frequency	100
Policy	DNN policy	MLP, 2 layers, 64 neurons

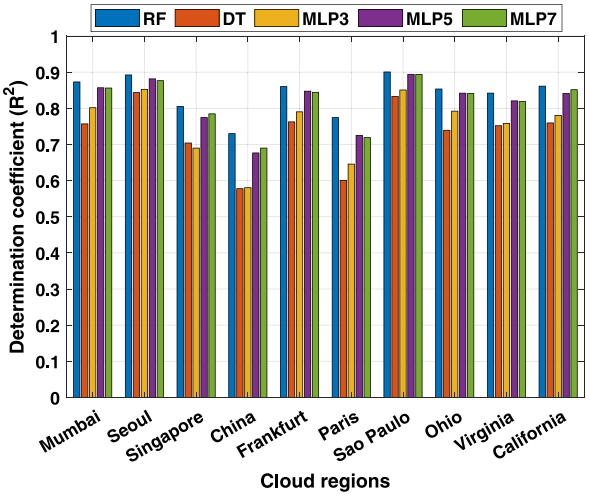


Fig. 7. Models testing: R^2 comparison.

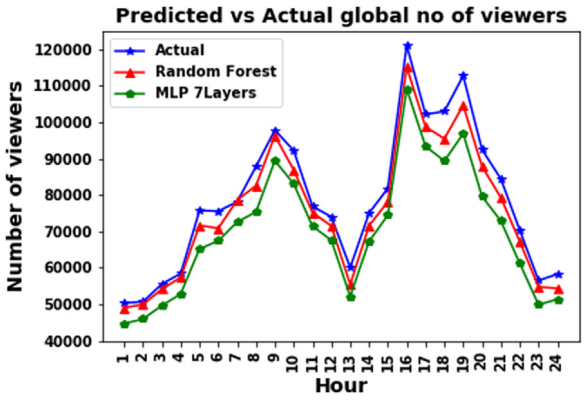


Fig. 8. Hourly actual vs predicted global number of viewers.

4.2. Simulation results

4.2.1. Performance of viewers prediction model

As we described previously in Section 3.2, we trained our data using three machine learning algorithms (RF, DT and MLP) and different hyper-parameters. We used only 225 thousand video records from the complete dataset to train our models. This reduced dataset was sufficient to achieve a satisfactory prediction. We divided the streams into 80% entries for the training task and 20% for the test and validation. Fig. 7 presents the results of testing models on the unseen data. Results show that the RF model outperforms MLP and DT in terms of R^2 by achieving 0.9 in Sao Paulo cloud site, 0.89 in Seoul region, and 0.87 and 0.86 in Mumbai and California regions, respectively. Furthermore, as we can see in the Figure, the prediction of the number of viewers is better in some regions, including Seoul and Sao Paulo, compared to other sites having lower accuracy such as China. The results show also that the DT algorithm presents the worst performance with an average R^2 equal to 0.73. Finally, the findings in Fig. 7 depict that increasing the number of layers of the MLP contributes to improve the prediction accuracy. However, since the improvement is minimal and due to the increasing complexity of MLP models, we did not test more than 7 layers. Next, we extend our experiments to perform hourly-based prediction for 24 h of July 3, 2018.

Fig. 8 shows the total predicted number of viewers globally at each hour compared to the actual total number.

Similarly, Fig. 9 presents a comparison between the predicted and the actual number of viewers in different cloud sites. We only presented RF and MLP-7-layers results as they performed better than other algorithms. These graphs further prove that the predicted popularity in different regions is close to the real one, particularly in Sao Paulo, Frankfurt, and Ohio. Finally, the results demonstrate that the RF algorithm achieves better accuracy than MLP. Therefore, we will adopt this model to predict the geo-distributed viewers, and results will be the input to the optimization problem and the RL approach.

4.2.2. Performance of RL-OPRA resource allocation

Fig. 10 illustrates the variation of cumulative rewards among training episodes. We trained the RL agent to 200 thousand episodes and we presented the average cumulative rewards for each 500 episodes.

We can notice that constraints are not respected at the beginning of the learning process. However, as the number of episodes increases, the number of respected constraints increases until reaching a stability after 25 thousand episodes, which confirms the convergence performance of our RL system. We note that the maximum cumulative rewards is equal to 10, if all constraints are respected for the $N = 10$ time-steps and the cost of the network is equal to 0. We can, also, see that before reaching 25 thousand episodes, the cumulative rewards learning increases fast. This is explained by the fact that, during this phase, the RL agent covers different allocation scenarios until finding the ones associated with optimal actions. Once actions with higher rewards are found, the agent obtains larger chances to increase its gain. Furthermore, Fig. 10 reveals that when the threshold is equal to 60, the learning process starts to be slow after 25 thousand episodes and converges after 150 thousand episodes. These results show that the threshold is very tight and finding the optimal allocation that respects the delay constraint and minimizes the cost is very challenging.

Fig. 11 shows the convergence of the cumulative cost penalties, when $\mathbb{D}$ is equal to 220 ms. We can see that minimizing the cost while being constrained by the latency is slowly learned and the penalties start to stabilize after 100 thousand episodes. This can be explained by the fact that this problem is NP-hard and the exploration space is large. However, when the system starts to discover optimal actions related to each received state, adequate decisions will be taken on the fly.

Fig. 12 shows the total cost incurred by our algorithm against the cost of the optimal solution and the Greedy Minimal Cost algorithm (GMC) [31], which considers constant resource provisioning. We evaluated our system under 3 latency thresholds: 60 ms, 120 ms, and 220 ms. The GMC algorithm is presented in the related work, Section 2. First, we can see that the cost incurred by the optimal solution, while having as input the actual number of viewers is very close to the results where the number of viewers is predicted. This proves again the robustness of our prediction. Next, we can see that our online reinforcement learning approach achieves near-optimal results and the same operational cost trend compared to the optimization. We can notice, also, that the RL-OPRA achieves lower costs in some scenarios, when the threshold is equal to 120 or 220. This can be explained by the fact that the accuracy of respecting the delay constraint is equal to 0.955. 0.957 and 0.85 for $\mathbb{D}$ equal to 60, 120 and 220, respectively. Fig. 14 shows the number of videos, where the system does not respect the average delay threshold while serving viewers. When checking these videos, we found that they correspond to contents that charge high amounts to be delivered with the right threshold. In such scenarios, the RL-OPRA allocates the live videos with much lower costs and a slightly higher delays than required. Meanwhile, the optimization serves all videos, while always respecting the required threshold.

When the threshold is equal to 60, the difference between the RL approach and the optimal one is more noticeable compared to other thresholds, which is explained by the challenging task (in terms of exploration) of achieving a delay close to 60 ms, while minimizing the cost. Moreover, we can notice that when the threshold is higher, the system incurs less operational costs. Therefore, the content provider can sacrifice in terms of cost to satisfy all viewers by serving them with low latencies or vice versa according to the application requirements. Finally, we compared our approach to GMC. We remind that GMC considers constant resources provisioning that we fixed in our simulation to be $AR = 50$ in each cloud site. Results show that GMC cost follows the trend of video broadcasting illustrated in Fig. 6, which is not the case of the optimal and the RL-based approach. This can be explained by the fact that GMC opted for a greedy solution, where the video is allocated in the site having minimum cost and able to serve all viewers while respecting the required threshold. If allocating the content in one site does not satisfy the requirements of the system, this algorithm keeps migrating copies of the video to the next cheapest cloud, until respecting the threshold. Such

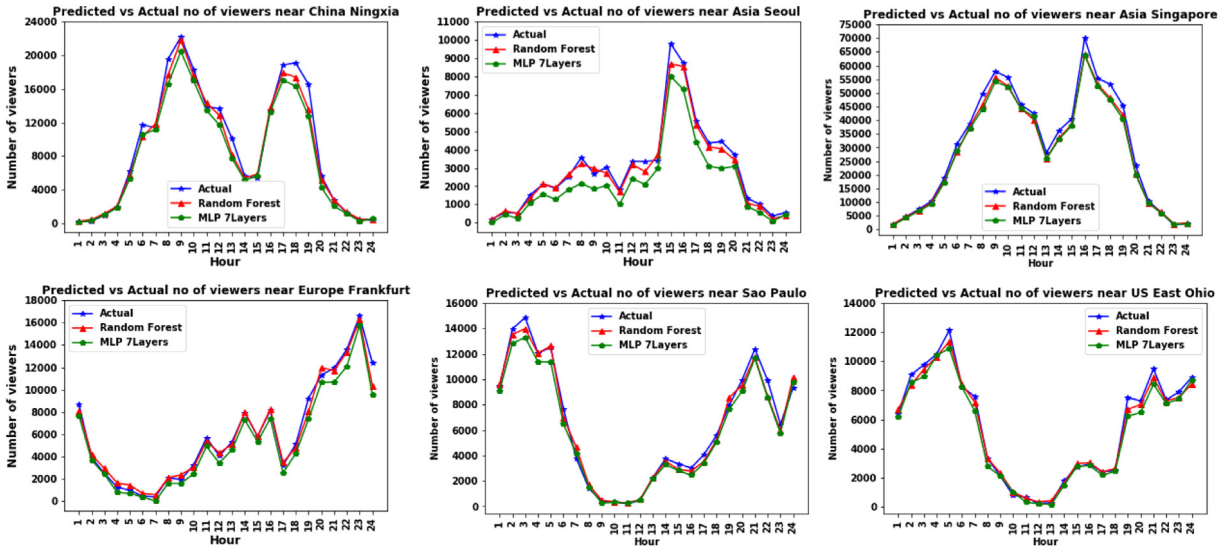


Fig. 9. Hourly actual vs predicted geo-distributed viewers.

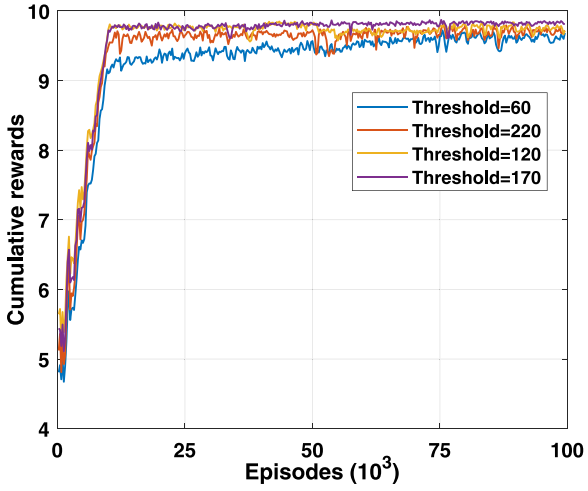


Fig. 10. Cumulative rewards vs. training episodes.

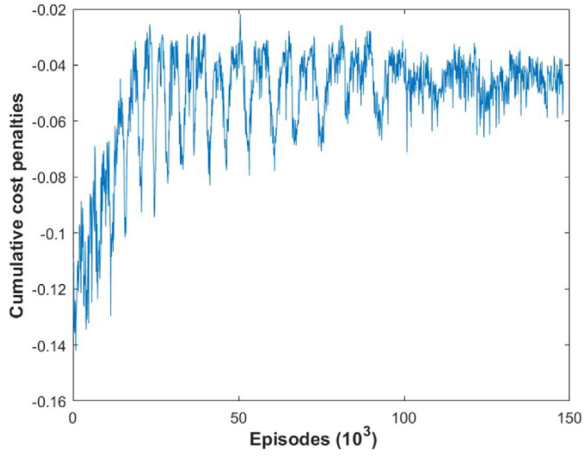


Fig. 11. Cumulative cost penalties (Threshold = 220 ms).

greedy solution applies a static strategy that does not take into consideration the distribution of viewers to establish a trade-off between the operational cost and the QoE, which makes the cost follow the trend of video broadcasting. As the optimization takes decisions based on an overview of the system state including costs, delays and number of potential geo-distributed viewers, it will present a non-static allocation. The same behavior is shown by our RL-based approach that learns the optimal policy and adapts to the dynamics of the crowdsourcing system. Therefore, compared to the greedy heuristic, our approach presents a lower operational cost.

Fig. 13 depicts our achieved hourly average delay compared to the optimal solution and the GMC approach. When the threshold is equal to 60 and 120 ms, our RL-OPRA accomplishes better perceived delays than the optimal solution. This can be interpreted by the fact that the predictive system does not present optimal allocation, which means videos can be allocated with slightly higher cost and lower delays. When the threshold is equal to 220 ms, the perceived delays are high, which is related to the impact of delivering a number of videos without respecting the threshold (see Fig. 14).

Regarding the computing complexity of our approach comparing to GMC, the DQN strategies depend, in general, on the adopted network, while heuristics depend on how they are designed. In our case, we adopted an MLP DNN network composed of 2 layers of 64 neurons, which is a light-weight network. Therefore, the complexity of our approach will be equal to $O(VNC)$, where V is the total number of videos, N is the number of cloud sites and C is the complexity of the DNN, which is negligible. Meanwhile, the complexity of the GMC approach is equal to $O(2VN)$. We can see that both complexities are similar, with a higher performance presented by our RL-OPRA approach. Furthermore, increasing the number of cloud sites N can add a complexity to the training phase, as the convergence and exploration speed is dependent on the size of the action set. In our case, the size of the action set is $N = 10$. In real world scenarios, the current number of AWS cloud sites is 22, the number of Facebook data centers is 12, 20 for Google and 54 for Microsoft, which is considered as acceptable for exploration complexity. We remind that the training is done offline or in parallel with online decisions.

To show how the RL-OPRA adapts to the system fluctuations, we trained the RL agent to 300 thousand episodes and we presented the average cumulative rewards/penalties for each 200 episodes. After 150 thousand episodes, we removed the cheapest data center and we opted for a geo-distributed cloud platform

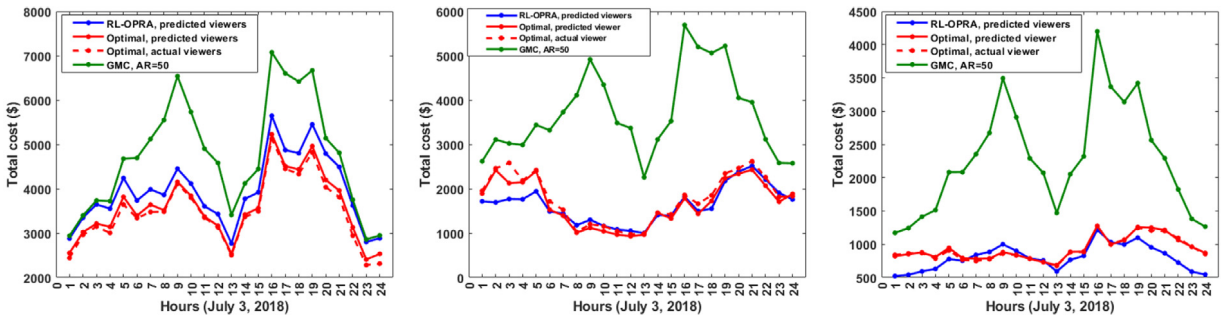


Fig. 12. Total system cost: RL-OPRA Vs Optimal Vs GMC: (1) Threshold = 60 ms, (2) Threshold = 120 ms, (3) Threshold = 220 ms.

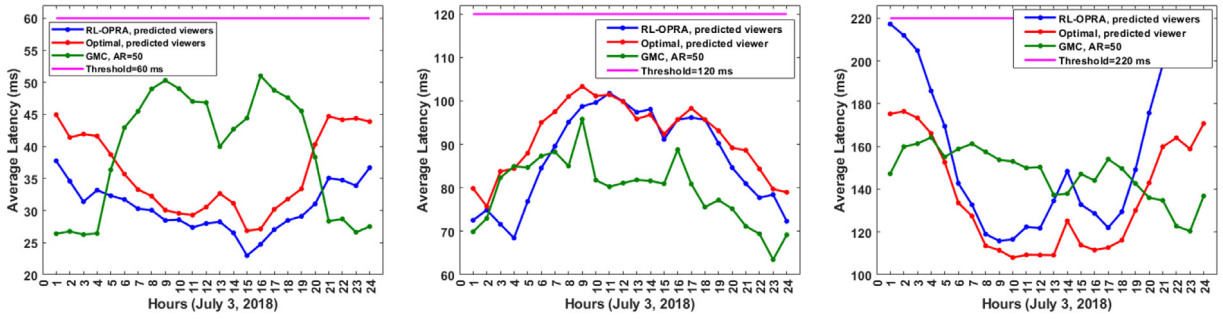


Fig. 13. Average delay: RL-OPRA Vs Optimal Vs GMC: (1) Threshold = 60 ms, (2) Threshold = 120 ms, (3) Threshold = 220 ms.

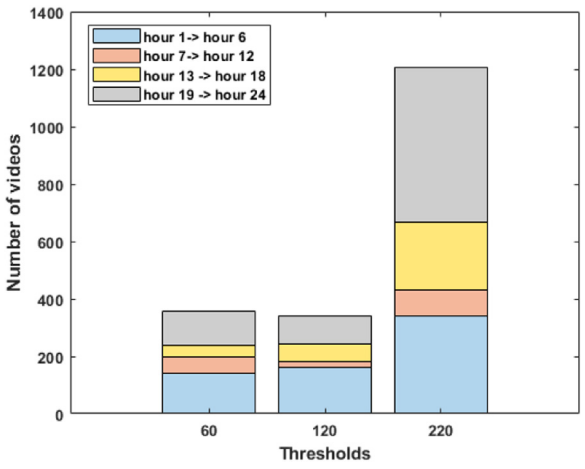


Fig. 14. Number of videos delivered without respecting thresholds over time.

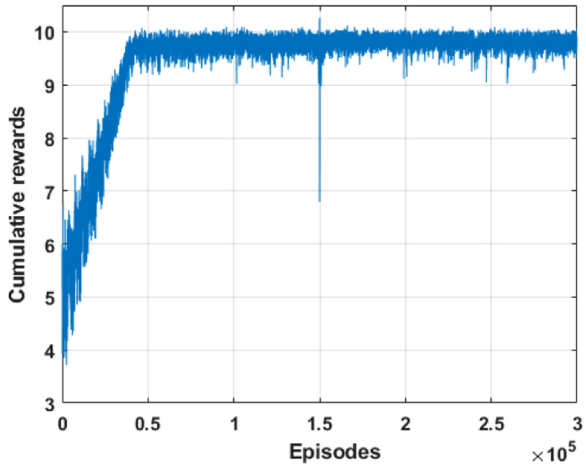


Fig. 15. Cumulative rewards while removing a cloud site at episode 150k.

of 9 sites. Figs. 15 and 16 illustrate the cumulative rewards and cumulative cost penalties, respectively. We can see in Fig. 15 that the cumulative rewards decreased, at episode 150k, because the system has learned to host videos at the cheapest region, which was disconnected from the platform. However, 200 episodes were enough for the RL agent to gain its performance and converge again. This rapid convergence shows that the RL-OPRA adapts to the environment changes easily and uses its past learning, specifically delays in different regions, to respect again the constraints. Fig. 16 shows that, when we removed the cheapest site, the cost penalties increased. However, the system directly converged, as it learned previously the costs in all data centers.

To conclude, our RL-OPRA system presented a good performance compared to heuristic based approaches, which is justified by the fact that DQN networks explore different scenarios until learning an optimum policy. The comparison to the ILP results proved that this policy is near-optimal. Additionally, the online popularity prediction of live videos helped the system to establish a proactive trade-off between operational costs and perceived delays. Finally, when the environment changes, RL-OPRA adapts easily to the new parameters, which is explained by the past learning.

5. Conclusion

In this paper, we studied the problem of offering a cost effective crowdsourcing live streaming services in a geo-distributed cloud platform while maximizing the viewers' QoE. Because of the dynamics of viewers behavior, we designed a predictive model that forecasts the potential number of viewers at each cloud site, based only on the content features. This predictive model is combined with our proposed RL-based resource allocation approach to present a video allocation solution, RL-OPRA, that performs well compared to the optimal results. Using extensive

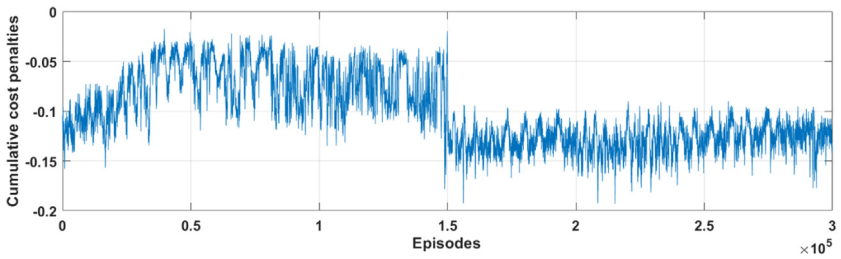


Fig. 16. Cumulative cost penalties while removing a cloud site at episode 150k.

simulations, we, also, illustrated that our RL-OPRA outperforms greedy-based heuristics and can adapt to the dynamics of the crowdsourcing system, through continuous learning. As a future work, we can consider serving users with their requested qualities as a QoE parameter. Furthermore, we can study the impact of having limited resources on the RL-approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

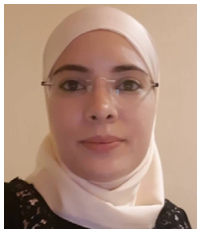
Acknowledgment

This work was supported by the Qatar Foundation.

References

[1] C. Zhang, J. Liu, H. Wang, Cloud-assisted crowdsourced livecast, ACM Trans. Multimed. Comput. Commun. Appl. 13 (3S) (2017) Art. no. 46.
[2] 25+ INcredible twitch statistics to know in 2020, 2020, URL https://leftronic.com/twitch-statistics/. (Accessed 19 January 2020).
[3] Number of monthly active facebook users worldwide as of 3rd quarter 2019, 2019, URL https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. (Accessed 19 January 2020).
[4] Facebook live: What you should know, 2017, URL https://telescope.tv/facebook-live-know/. (Accessed 19 January 2020).
[5] Q. He, J. Liu, C. Wang, B. Li, Coping with heterogeneous video contributors and viewers in crowdsourced live streaming: A cloud-based approach, IEEE Trans. Multimed. 18 (5) (2016) 916–928.
[6] F. Haouari, E. Baccour, A. Erbad, A. Mohamed, M. Guizani, QoE-Aware Resource Allocation for Crowdsourced Live Streaming: A Machine Learning Approach, in: ICC 2019-2019 IEEE International Conference on Communications, ICC, Shanghai, China, 2019, pp. 1–6.
[7] A. Ben Said, A. Erradi, A.G. Neiat, A. Bouguettaya, A deep learning rowdsourced services, Mobile Netw. Appl. 24 (3) (2018) 1120–1133.
[8] A. Ben Said, A. Erradi, A probabilistic approach for maximizing travel journey wifi coverage using mobile crowdsourced services, IEEE Access 7 (2019) 82297–82307.
[9] Vishal Sharma, Ilsun You, Dushantha Nalin K. Jayakody, Mohammed Atiquzzaman, Cooperative trust relaying and privacy preservation via edge-crowdsourcing in social Internet of Things, Future Gener. Comput. Syst. 92 (2019) 758–776.
[10] W. Xiao, W. Bao, X. Zhu, L. Liu, Cost-aware big data processing across geo-distributed datacenters, IEEE Trans. Parallel Distrib. Syst. 28 (11) (2017) 3114–3127.
[11] Chongwu Dong, et al., Joint optimization of data-center selection and video-streaming distribution for crowdsourced live streaming in a geo-distributed cloud platform, IEEE Trans. Netw. Serv. Manag. (2019) 729–742.
[12] Lei Wei, et al., Qos-aware resource allocation for video transcoding in clouds, IEEE Trans. Circuits Syst. Video Technol. 27 (1) (2017) 49–61.
[13] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, F.C.M. Lau, Scaling social media applications into geo-distributed clouds, IEEE/ACM Trans. Netw. 23 (3) (2015) 689–702.
[14] X. Wang, B. Fang, H. Zhang, X. Wang, A Dynamic Model on News Popularity Prediction in Online Social Networks, in: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC, Chengdu, China, 2019, pp. 847–851.
[15] Yen-Liang Chen, Chia-Ling Chang, Early prediction of the future popularity of uploaded videos, Expert Syst. Appl. 133 (2019) 59–74.
[16] J. Liu, S.G. Rao, B. Li, H. Zhang, Opportunities and challenges of peer-to-peer internet video broadcast, Proc. IEEE 96 (1) (2008) 11–24.
[17] V.K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, Z. Zhang, Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery, in: Proc. IEEE INFOCOM, 2012.
[18] F. Wang, J. Liu, M. Chen, CALMS: Migration towards CloudAssisted Live Media Streaming, in: Proc. IEEE INFOCOM, 2012.
[19] P. Simoens, et al., Scalable crowd-sourcing of video from mobile devices, in: Proc. ACM 11th Annu. Int. Conf. Mobile Syst. Appl. Services, Taipei, Taiwan, 2013, pp. 139–152.
[20] C. Zhang, J. Liu, On crowdsourced interactive live streaming: A twitch. TV-based measurement study, in: Proc. 25th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video, Portland, OR, USA, 2015, pp. 55–60.
[21] K. Pires, G. Simon, Youtube live and twitch: A tour of usergenerated live streaming systems, in: Proceedings of the 6th ACM Multimedia Systems Conference, MMSys '15, ACM, 2015, pp. 225–230, ser..
[22] C. Zhang, J. Liu, H. Wang, Towards hybrid cloud-assisted crowdsourced live streaming: Measurement and analysis, in: Proc. ACM 26th Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video, Klagenfurt, Austria, 2016, Art. no. 1.
[23] R. Shea, D. Fu, J. Liu, Towards bridging online game playing and live broadcasting: Design and optimization, in: Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '15, ACM, 2015, pp. 61–66, ser..
[24] T. Smith, M. Obrist, P. Wright, Live-streaming changes the (video) game, in: Proceedings of the 11th European Conference on Interactive TV and Video, EuroITV '13, ACM, 2013, pp. 131–138, ser..
[25] Emna Baccour, Aiman Erbad, Kashif Bilal, Amr Mohamed, Mohsen Guizani, Mounir Hamdi, FacebookVideoLive18: A live video streaming dataset for streams metadata and online viewers locations, 2020, arXiv:2003.10820.
[26] Amazon EC2 reserved instances pricing, 2020, URL https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/. (Accessed 19 January 2020).
[27] Fei Chen, et al., Cloud-assisted live streaming for crowdsourced multimedia content, IEEE Trans. Multimed. 17 (9) (2015) 1471–1483.
[28] F. Wang, J. Liu, M. Chen, H. Wang, Migration towards cloud assisted live media streaming, IEEE/ACM Trans. Netw. 24 (1) (2016) 272–282.
[29] W. Xiao, et al., Dynamic request redirection and resource provisioning for cloud-based video services under heterogeneous environment, IEEE Trans. Parallel Distrib. Syst. 27 (7) (2016) 1954–1967.
[30] X. Qiu, H. Li, C. Wu, Z. Li, F.C. Lau, Cost-minimizing dynamic migration of content distribution services into hybrid clouds, IEEE Trans. Parallel Distrib. Syst. 26 (12) (2015) 3330–3345.
[31] K. Bilal, A. Erbad, M. Hefeeda, QoE-aware distributed cloud-based live streaming of multi-sourced multiview videos, J. Netw. Comput. Appl. 120 (2018) 130–144.
[32] A.A. Haghighi, S.S. Heydari, S. Shahbazpanahi, Dynamic QoS aware resource assignment in cloud-based content-delivery networks, IEEE Access 6 (2018) 2298–2309.
[33] F. Haouari, E. Baccour, A. Erbad, A. Mohamed, M. Guizani, Transcoding Resources Forecasting and Reservation for Crowdsourced Live Streaming, in: 2019 IEEE Global Communications Conference, GLOBECOM, Waikoloa, HI, USA, 2019, pp. 1–7.
[34] X. Yuan, M. Sun, Q. Fang, C. Du, DLECP: A Dynamic Learning-based Edge Cloud Placement Framework for Mobile Cloud Computing, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, Paris, France, 2019, pp. 1035–1036.
[35] S. Ouyang, C. Li, X. Li, A peek into the future: Predicting the popularity of online videos, IEEE Access 4 (2016) 3026–3033.
[36] C. Li, J. Liu, S. Ouyang, Characterizing and predicting the popularity of online videos, IEEE Access 4 (2016) 1630–1641.
[37] J. Wu, Y. Zhou, D.M. Chiu, Z. Zhu, Modeling dynamics of online video popularity, IEEE Trans. Multimed. 18 (9) (2016) 1882–1895.
[38] T. Trzciński, P. Rokita, Predicting popularity of online videos using support vector regression, IEEE Trans. Multimed. 19 (11) (2017) 2561–2570.
[39] A. Bielski, T. Trzcinski, Understanding multimodal popularity prediction of social media videos with self-attention, IEEE Access 6 (2018) 74277–74287.

[40] H. Jeon, W. Seo, E. Lucy Park, Sungchul Choi, Hybrid machine learning approach to popularity prediction of newly released contents for online video streaming service, 2019, arXiv:1901.09613.
[41] Amazon EC2, 2019, URL https://aws.amazon.com/fr/ec2/. (Accessed 01 November 2019).
[42] Amazon S3, 2019, URL https://aws.amazon.com/fr/s3/. (Accessed 01 November 2019).
[43] Global Infrastructure, 2019, URL https://aws.amazon.com/about-aws/global-infrastructure/. (Accessed 19 January 2020).
[44] K. Mnih V. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjel, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[45] Global ping statistics, 2020, URL https://wondernetwork.com/pings. (Accessed 01 November 2019).



Emna Baccour received the Ph.D. degree in computer Science from the University of Burgundy, France, in 2017. She was a postdoctoral fellow at Qatar University on a project covering the interconnection networks for massive data centers and then on a project covering video caching and processing in mobile edge computing networks. She currently holds a postdoctoral position at Hamad Ben Khalifa University. Her research interests include data center networks, cloud computing, green computing and software defined networks as well as distributed systems. She is also interested in edge networks and mobile edge caching and computing.

Aiman Erbad is an Associate Professor at the College of Science and Engineering at Hamad Bin Khalifa University (HBKU). Dr. Erbad obtained a Ph.D. in Computer Science from the University of British Columbia (Canada), and a Master of Computer Science in Embedded Systems and Robotics from the University of Essex (UK). Dr. Erbad received the Platinum award from H.H. The Emir Sheikh Tamim bin Hamad Al Thani at the Education Excellence Day 2013 (Ph.D. category). Dr. Erbad received the 2020 best research paper award from the Computer Communications journal, IWCMC 2019 best paper award, and IEEE CCWC 2017 best paper award. Dr. Erbad is an editor in KSII Transactions on Internet and Information Systems and was a guest editor in IEEE Networks. Dr. Erbad research interests span cloud computing, edge computing, IoT, private and secure networks, and multimedia systems.

Amr Mohamed (S' 00, M' 06, SM' 14) received his M.S. and Ph.D. in electrical and computer engineering from the University of British Columbia, Vancouver, Canada, in 2001, and 2006 respectively. He has worked as an advisory IT specialist in IBM Innovation Centre in Vancouver from 1998 to 2007, taking a leadership role in systems development for vertical industries. He is currently a professor in the college of engineering at Qatar University and the director of the Cisco Regional Academy. He has over 25 years of experience in wireless networking research and industrial systems development. He holds 3 awards from IBM Canada for his achievements and leadership, and 4 best paper awards from IEEE conferences. His research interests include wireless networking, and edge computing for IoT applications. Dr. Amr Mohamed has authored or co-authored over 160 refereed journal and conference papers, textbook, and book chapters in reputable international journals, and conferences. He is serving as a technical editor for the journal of internet technology and the international journal of sensor networks. He has served as a technical program committee (TPC) co-chair for workshops in IEEE WCNC'16. He has served as a co-chair for technical symposia of international conferences, including Globecom'16, Crowncom'15, AICCSA'14, IEEE WLN'11, and IEEE ICT'10. He has served on the organization committee of many other international conferences as a TPC member, including the IEEE ICC, GLOBECOM, WCNC, LCN and PIMRC, and a technical reviewer for many international IEEE, ACM, Elsevier, Springer, and Wiley journals.

Fatima Haouari received the B.Sc. and MSc. degree in computer science (with distinction) from Qatar University. She is currently a Ph.D. student and a Research Assistant at Qatar University. Her research interests span cloud computing, crowdsourced multimedia, and machine learning. She also interested in fog/edge computing and distributed systems.

Mohsen Guizani (S'85–M'89–SM'99–F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering, the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor at the Computer Science and Engineering Department in Qatar University, Qatar. Previously, he served in different academic and administrative positions at the University of Idaho, Western Michigan University, University of West Florida, University of Missouri-Kansas City, University of Colorado-Boulder, and Syracuse University. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is currently the Editor-in-Chief of the IEEE Network Magazine, serves on the editorial boards of several international technical journals and the Founder and Editor-in-Chief of Wireless Communications and Mobile Computing journal (Wiley). He is the author of nine books and more than 500 publications in refereed journals and conferences. He guest edited a number of special issues in IEEE journals and magazines. He also served as a member, Chair, and General Chair of a number of international conferences. Throughout his career, he received three teaching awards and four research awards. He also received the 2017 IEEE Communications Society WTC Recognition Award as well as the 2018 Ad hoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad hoc Sensor networks. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer. He is a Fellow of IEEE and a Senior Member of ACM.

Mounir Hamdi received the B.S. degree (Hons.) in electrical engineering (computer engineering) from the University of Louisiana, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from the University of Pittsburgh, in 1987 and 1991, respectively. He was a Chair Professor and a Founding Member of The Hong Kong University of Science and Technology (HKUST), where he was the Head of the Department of Computer Science and Engineering. From 1999 to 2000, he held visiting professor positions at Stanford University and the Swiss Federal Institute of Technology. He is currently the Founding Dean of the College of Science and Engineering, Hamad Bin Khalifa University (HBKU). His area of research is in high-speed wired/wireless networking, in which he has published more than 360 publications, graduated more 50 M.S./Ph.D. students, and awarded numerous research grants. In addition, he has frequently consulted for companies and governmental organizations in the USA, Europe, and Asia. He is a Fellow of the IEEE for his contributions to design and analysis of high-speed packet switching, which is the highest research distinction bestowed by IEEE. He is also a frequent keynote speaker in international conferences and forums. He is/was on the editorial board of more than ten prestigious journals and magazines. He has chaired more than 20 international conferences and workshops. In addition to his commitment to research and academic/professional service, he is also a dedicated teacher and a quality assurance educator. He received the Best 10 Lecturer Award and the Distinguished Engineering Teaching Appreciation Award from HKUST. He is frequently involved in higher education quality assurance activities as well as engineering programs accreditation all over the world.