Cost aware Caching and Streaming Scheduling for Efficient Cloud based TV

Zinelaabidine Nadir[1,2], Miloud Bagaa[2], and Tarik Taleb[2]

[1] University of Laghouat, Algeria, z.nadir@lagh-univ.dz
[2] Aalto University, Finland, firstname.lastname@aalto.fi

***Abstract*—Internet Protocol Television (IPTV) has become widely used to deliver TV channels over the Internet. Tremendous efforts have been carried out for making IPTV services an alternative to traditional TV by offering low-cost TV channels. The cloud network offers many advantages that give more flexibility for sharing the content and reducing the cost for end-users. In this paper, we explore the strength of cloud by allowing different users to create cost-efficient TV channels on top of the cloud. The proposed algorithm reduces the cost by exploring the shared content in the cloud network. The simultaneous streaming of the content shared among different channels will reduce the number of streams in the network, and consequently the otherwise incurred cost. This can be achieved through a smart scheduling mechanism that schedules the streaming of the same video to a large number of channels at the same time. The obtained results prove the efficiency of the proposed solution in terms of cost efficiency.**

I. Introduction

Multimedia streaming services define the de-facto service among an ever-growing community of digital content consumers, be they mobile or not [1], [2]. The Internet Protocol TV (IPTV) services define an imporant use case of multimedia streaming services. The purpose of IPTV is to provide a cost-efficient alternative to the traditional TV system, allowing the distribution of different video content to end-users over IP networks and reducing significantly the cost of creating and managing different TV channels. Aggarwal et al. [3] have shown that the cost of IPTV services would be reduced further leveraging cloud infrastructure. Indeed, through the usage of cloud, a simple user becomes able to hold different channels that serve multiple social communities. In contrast to traditional TV channels that would need dedicated equipment and technical stuff for managing individual channels, IPTV channels on top of the cloud will enable one simple user (channel owner) to manage many channels from home without the need for neither dedicated equipments nor technical stuff [4].

Enabling users to create Over-The-Top (OTT) TV channels on top of the cloud will make the content more customized for their social communities, defining the channels' audiences [5], [6]. The small number of subscribers gives to the channel owner the possibility to stream content customized according to the profiles of his channel's subscribers. In fact, this is motivated by recent studies that showed the positive impact of users' social interactions on TV channels' experience [7]–[9], whereby TV programs could be customized based on, among many factors, social interactions and comments of channel subscribers. In fact, as shown in Figure 1, a typical TV channel requires: i) an appropriate multimedia delivery infrastructure, ii) content to deliver, and iii) an audience for the channel. The advances in the cloud computing technology facilitates the creation of media delivery infrastructure along with their enabling techniques such as transcoding on the fly [10]–[12]. This infrastructure will be used to store, cache videos on one of its servers and to stream them to channels' audience. The channel content could be self-made videos, content from online video-sharing websites (e.g., Youtube, dailymotion, and Vimeo) or videos from other real TV broadcasting channels. The channel's audience consists of those who are invited using different social media networks (e.g., Facebook, twitter, and LinkedIn) to watch these videos together on a specific schedule set by the channel owner. The main function of the system, shown in Figure 1, is to allow users to create OTT TV channels and to manage their content and their audience [4]. Quite simply, using such system, the channel owner can create his own channel by uploading one or multiple videos to the system and share them with his subscribers and friends, in the form of TV program, to watch them as a specific schedule.

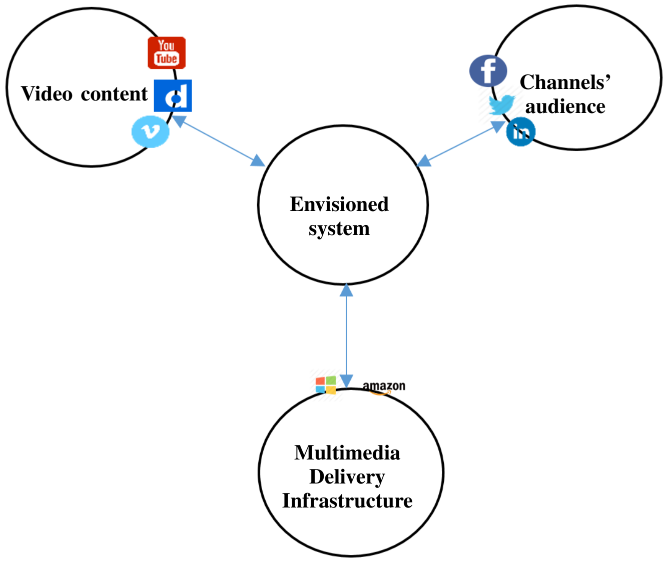

Fig. 1. Key elements of the envisioned system.

Nowadays, most of cloud providers adopt pay-as-you-go model, whereby a user will pay only for the resources that he indeed used, including CPU, Memory, Storage and traffic communication [13]. The videos of a channel will be cached in servers that guarantee the Quality of Service (QoS) for its subscribers. Thus, the channel owner will be charged for the time of caching and streaming of those videos from those servers. In order to reduce the cost for the channel owner, the caching and streaming time of different videos should be minimized as much as possible. Indeed, the aim is to make the caching time fits, as much as possible, the streaming time.

The purpose of the system, presented in this paper, is to devise an efficient algorithm for $i)$ scheduling the playback of different content over multiple channels, serving a geographically distributed group of subscribers, and $ii)$ accordingly caching the content across an underlying content delivery network in a way that minimizes the cost incurred with both caching and streaming of each video while guarantying the desired QoS. For this reason, the proposed algorithm would specify the streaming time and the locations of different copies (caches) of that video for ensuring the QoS with the minimal cost. Herein, the location of a cache content means the server where the content cache should be stored. The remainder of the paper is organized as follows. Some related work are presented in Section II. In Section III, we formulate the target problem. The proposed solution is presented in Section IV. The performance evaluation of the proposed solution is conducted in Section V and the simulation results are discussed therein. Finally, the paper concludes in Section VI.

II. Related work

Ensuring QoS for multimedia content requires high network bandwidth between the server that holds that content and different clients [1]. Placing the content in far away servers may downgrade the Quality of Experience (QoE) [14], [15]. On the other hand, duplicating and placing the content near to the clients can be an expensive process. In order to overcome this problem, smart caching techniques, combined with Information (or Content) Centric Networking (ICN/CCN) techniques, have been proposed [16], [17]. These techniques cache the multimedia content according to their geographic popularity. Indeed, each content video would be duplicated and cached only in the regions where that video is popular. Along with the emergence of online social networks (OSN), the management of multimedia caching has become more challenging [18]. Indeed, the users community of a multimedia content can be distributed over the world. In order to overcome this problem, the authors in [18] suggested a new caching strategy whereby a video is cached in a region only if the video popularity is higher than a predefined threshold.

Cloud computing has been gaining tremedous potential, impacting different arenas including mobile telecom [19]. It brings more advantages when compared to the traditional architectures, and that is in terms of elasticity and resource provisioning with minimal cost. Many techniques have been proposed for managing video caching on top of the cloud. The caching techniques aim to reduce the cost while guarantying the QoS. These techniques utilize social features to cache the content with QoS. Pujol et al. [20] have proposed an approach that caches videos of a user in the same server with his friends' videos. However, if videos are assigned to servers based only on their social information, servers with popular videos could be accessed more frequently than the ones with unpopular videos. Therefore, some servers will be more overloaded than others. To overcome this problem, Cheng et al [21] have proposed an approach that takes into account the video popularity and social information for caching videos contents. Meanwhile, Wu et al. [22] utilized a geo-distributed cloud to support large scale social media streaming applications. Wang et al. [23] have proposed an approach that allows the caching servers to adjust their resources according to the users' demands. This approach enables caching servers to adapt their resource capacity to meet with the temporal and the spatial dynamics of users access. Novel concepts, such as dynamic Service Function Chaining (SFC) coupled with the benefits of Netowrk Function Virtualization (NFV) and cloud computing, have been also explored to cope with social media applications that involve frequent delivery of videos among a large number of social networks [24].

In contrast to the aforementioned works, the proposed solution, in this paper, aims to reduce the caching and streaming costs of OTT TV channels. The caching and streaming costs will be reduced by promoting the sharing of the same video streams among different subscribers of different TV channels. Moreover, the proposed solution considers both the profiles of the channels' subscribers and the access patterns of channels' videos. The former is used to decide where to cache these videos, whereas the latter is used to decide when to cache them.

III. Problem formulation and envisioned architecture

A. Envisioned architecture

Figure 2 depicts a general overview of the system architecture and its principal components. The envisioned architecture consists of the main system server and several cache/stream servers. The system server is responsible for managing and scheduling different multimedia content in the cloud. The cache/stream servers are geographically distributed as part of the cloud services (e.g., Amazon and Microsoft Azure), where each server plays the role of a cache and/or streaming server. These servers will serve many viewers of different TV channels. We assume that these servers have no limited capacity of storage. We also assume that these servers can support any streaming rates and have no limitation in terms of number of simultaneous streams they can support. In this paper, we propose an algorithm that will be executed on top of the system server for managing and scheduling content in the programs of different TV channels.

The contents of TV channels over the cloud are usually distributed on different clouds, as well as the channels' subscribers would be. The channel's subscribers can be located

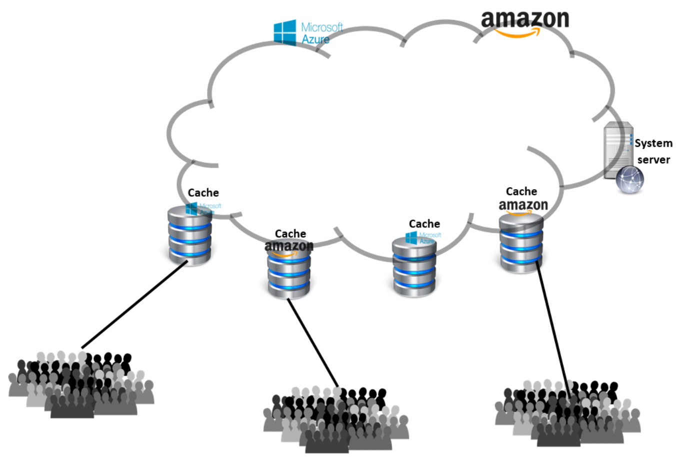

Fig. 2. The envisioned architecture.

in many regions/countries in the world. They consist of those who are invited by the channel owner using different social media networks to watch his channel. The main function of the system server is to allow users to manage their channels and that includes: i) Creating a channel, uploading videos, and inviting users to that channel; ii) Choosing the appropriate streaming servers among the available ones for each video content; iii) Scheduling different videos content by specifying the exact time where they should be played. Moreover, the proposed framework aims to guarantee the QoS for all subscribers from any region. According to the end-to-end delay between a server and channel's subscribers, the server may guarantee or not the QoS. For this reason, the videos of a TV channel should be placed in a server by taking into account the end-to-end delay between the channel subscribers and that server.

B. Network model and problem formulation

The goal of this paper is to reduce the cost for deploying different TV channels on top of the cloud while guaranteeing QoS. Therefore, the aim is to reduce the number of streams and caching time for each video. In order to reduce the caching time, the videos should be cached only during their streaming time. To reduce further the cost, the number of videos to be streamed at approximately the same time over different channels should be increased. If multiple channels schedule in their program the same video at nearly the same time and share the same cache content, in the same sever, they will share also the caching cost among them. Moreover, they will also share the streaming cost. Indeed, if two channels do not succeed to schedule the playback of a given video at the same time, then the number of streams of that server becomes two even if they share the same cache content. Herein, each channel will pay the complete cost of one stream. In this case, the benefit of both channels is the reduction of caching cost.

We assume that the network consists of a set of streaming servers S and a set of users U. Let C denote the set of TV channels, each channel i ($i \in C$) may contain a set of scheduled videos that will be cached in the streaming servers before their streaming times. Videos of a channel i will be streamed to its subscribers U_i, where $U_i \subset U$ is the subscribers of channel i. A user $u \in U$ can be subscribed to one or multiple channels. Let R_i be the set of regions of channel i. Let R denote the set of regions in the network. Formally, $R = \bigcup_{i \in \mathcal{C}} R_i$. A user of channel i is also supposed to be in one region $r/r \in R_i$ at a given time. In this paper, we assume that a server $l \in S$ guarantees the QoS for a region $r \in R$ iff the end-to-end delay between that server and a user in r is lower than a predefined threshold τ. The system server should guarantee QoS for all users in different regions when scheduling videos in channel programs and caching videos.

TABLE I
DESCRIPTION OF NOTATIONS AND VARIABLES USED IN THE MODEL.

variable	description
$\mathcal{C}$	The set of channels.
V	The set of videos.
V_i	The set of videos that are already scheduled in channel i.
$\mathcal{U}$	The set of users in the network.
$\mathcal{U}_j$	A set of subscribers of channel j.
x	A user, $x \in \mathcal{U}$.
d_j	The duration of video j.
$\mathcal{S}$	The set of servers.
$\mathcal{R}$	The system set of regions.
$\mathcal{R}_i$	Set of regions of channel i.
$t_{i,j}$	The playback time of video j in channel i.
Υ_i	Set of videos that a user wants to add to his channel i.
$s_{i,j}$	An integer variable that indicates the schedule time of video j in channel j.
$x_{i,j,l}$	A Boolean variable that equals to 1 if video j in channel i will be cached in server l; 0 otherwise.
$y_{i,i',j,l}$	A Boolean variable that equals to 1 if channels i and i' succeed to schedule video j at the same time and from the same server l; 0 otherwise.
$\eta_{j,l}$	Number of streams of video j in server l.
$e2e_{r,l}$	The end-to-end delay between server l and users in region r.
τ	End-to-end threshold that is used for ensuring QoS. If a delay between users in a region and a server is lower than τ, then QoS will be ensured.

The different notations and variables used in this paper are summarized in Table I. Let Υ_i denote the set of videos of a channel $i \in \mathcal{C}$. Administrator of each channel i should ensure QoS for its subscribers with minimal cost. The scheduling of a channel means specifying the streaming time and the caching server of each video scheduled in the program of that channel. As aforementioned, the proposed framework explores the schedules of videos in other channels for reducing the caching and streaming costs. Let C_j denote the set of channels where a video j is already scheduled. Note that a video will not be cached in a server until its streaming time is approached. For this reason, while the streaming time of videos in scheduled channel C_j should not be changed, the caching servers can be updated.

For example, let's assume that a video v is already scheduled in a channel j. Let t and l_1 denote, respectively, the scheduling time of v and the server from where v will be streamed. Lets assume that the owner of another channel i wants to update his channel with that video. If the server l_1 does not guarantee QoS for the users of channel i, then v should be cached in another server that ensures QoS for those users. Thus, channels i and j will not share the same video stream. In order to reduce the streaming and caching time, an alternative strategy is adopted in this paper. This strategy consists of changing the caching servers of already scheduled channels. In this example, if we cache v in another server l_2 that guarantees QoS for the subscribers of both channels. In this case, both channels will succeed to stream v to their subscribers using only a single stream and from the same server. This feature will enable the reduction of the costs of both caching and streaming of the videos already scheduled in channels by changing their caching servers whenever the program of a new channel gets updated with one of these videos.

IV. Proposed Solution

In the following section we will formulate our problem as a Mixed Integer Linear Programming (MILP) problem. As mentioned before, we are interested in the optimal placement of video content and scheduling different channels in order to reduce the caching and streaming costs. This will be achieved by reducing the number of effective streams of each video j. Channels that succeed to cache video j in the same cache server at the same time, they will share also the caching and streaming costs. Otherwise, if these channels do not succeed to cache that video in the same cache server and/or stream it simultaneously, each channel will pay, separately, the cost of caching and streaming of that video. Let i denote the active channel that will be updated with a set of videos Υ_i. To reduce the cost, the number of streams should be reduced for each video $j \in \Upsilon_i$. For each video j, we need to find a schedule in that channel and a server, where the video should be cached. Let $s_{i,j}$ be a variable that indicates the scheduling time of video j in channel i. Let $x_{i,j,l}$ be a boolean variable that takes the value 1 if video j is cached in server l, otherwise 0.

Each video j could potentially be scheduled in other channels. Let $t_{i',j}$ be the playback time of video j in Channel i' ($i' \in C_j$; $i \neq i'$). To reduce the number of streams, Channel i needs to: 1) schedule video j at the same time with as many channels C_j as possible; 2) cache that video in the same cache server with these channels. The latter will reallocate the caching servers of video j for each channel C_j in order to create the minimum number of streams of that video. If we do not succeed to schedule video j at the same time with other channels C_j, the owner of channel i will pay for a complete stream of video j.

$$\mathbf{min} \sum_{\substack{j \in \Upsilon_i \\ l \in S}} \eta_{j,l} \tag{1}$$

Subject to:

$$\forall j \in \Upsilon_i : \sum_{l \in S} x_{i,j,l} = 1 \tag{2}$$

$$\forall j \in \Upsilon_i,\ \forall i' \in C_j,\ \forall j' \in V_{i'} : \sum_{l \in S} x_{i',j',l} = 1 \tag{3}$$

$\forall j \in \Upsilon_i,\ \forall j' \in V_i$

$$s_{i,j} \geq t_{i,j'} + d_{j'} + a_{i,j,j'} \times M \tag{4}$$

$$s_{i,j} \leq t_{i,j'} - d_j + (1 - a_{i,j,j'}) \times M \tag{5}$$

$\forall j, j' \in \Upsilon_i,\ j \neq j'$

$$s_{i,j} \geq s_{i,j'} + d_{j'} + a'_{i,j,j'} \times M \tag{6}$$

$$s_{i,j} \leq s_{i,j} - d_j + (1 - a'_{i,j,j'}) \times M \tag{7}$$

$\forall j \in \Upsilon_i,\ \forall i' \in C_j,\ \forall l \in S$

$$|t_{i',j} - s_{i,j}| \leq (1 - y_{i,i',j,l}) \times M \tag{8}$$

$$y_{i,i',j,l} \leq x_{i,j,l} \tag{9}$$

$$y_{i,i',j,l} \leq x_{i',j,l} \tag{10}$$

$\forall j \in \Upsilon_i,\ \forall i', i'' \in C_j,\ i' \neq i'',\ \forall l \in S$

$$|t_{i',j} - t_{i'',j}| \leq (1 - y_{i',i'',j,l}) \times M \tag{11}$$

$$y_{i',i'',j,l} \leq x_{i',j,l} \tag{12}$$

$$y_{i',i'',j,l} \leq x_{i'',j,l} \tag{13}$$

$\forall j \in \Upsilon_i,\ \forall l \in S, \forall r \in R_i$

$$x_{i,j,l} \times e2e_{r,l} \leq \tau \tag{14}$$

$\forall j \in \Upsilon_i,\ \forall l \in S, \forall i' \in C_j, \forall r \in R_{i'}, i' \neq i$

$$x_{i',j,l} \times e2e_{r,l} \leq \tau \tag{15}$$

$\forall j \in \Upsilon_i,\ \forall l \in S$

$$\rho_{i,j,l} - \sum_{\forall i' \in C_j} y_{i,i',j,l} \leq 0 \tag{16}$$

$\forall j \in \Upsilon_i,\ \forall i' \in C_j,\ \forall l \in S$

$$\rho_{i,j,l} - y_{i,i',j,l} \geq 0 \tag{17}$$

$\forall j \in \Upsilon_i,\ \forall i' \in C_j,\ \forall l \in S$

$$\rho_{i',j,l} - \sum_{\substack{\forall i'' \in C_j \\ i'' > i'}} y_{i',i'',j,l} \leq 0 \tag{18}$$

$\forall j \in \Upsilon_i,\ \forall i', i'' \in C_j,\ i'' > i'',\ \forall l \in S$

$$\rho_{i',j,l} - y_{i',i'',j,l} \geq 0 \tag{19}$$

$\forall j \in \Upsilon_i,\ \forall l \in S$

$$\eta_{j,l} - \left(x_{i,j,l} + \sum_{\substack{\forall i' \in C_j \\ i' > i}} x_{i',j,l} \right) - \left(\rho_{i,j,l} + \sum_{\substack{\forall i' \in C_j \\ i' > i}} x_{i',j,l} - \rho_{i',j,l} \right) = 0 \quad (20)$$

In the envisioned optimization problem, we use the following notations: 1) variable $\eta_{j,l}$ defining the number of streams generated for each video j in server l; 2) variable i representing the current channel, where videos Υ_j will be added; 3) V_i denoting the set of videos that are already scheduled in channel i. The objective function aims to minimize the number of streams of each video j in all servers l, where the video will be cached. Meanwhile, the constraints are used to ensure the following conditions: constraints (2) and (3) ensure that each channel should cache each video in only one server. Constraints (4)-(7) ensure that the scheduled time of each video should not overlap with any other video in channel i. Constraints (4) and (5) ensure that the scheduled time of each video j ($\forall j \in \Upsilon_i$) should not overlap with the current channel's videos V_i. This means that two videos should not have their playing times overlap in channel i. Whereas, constraints (6) and (7) ensure that the playing times of videos from Υ_i should not overlap with each other.

Constraints (8)-(13) ensure that videos that have the same scheduled time should be cached in the same server to reduce the number of streams. Constraints (8)-(10) ensure that if channels i and i' scheduled video j at the same time, then they should use the same cache server l for reducing the streaming cost. In the same manner, Constraints (11)-(13) ensure that if video j in channels i' and i'' is scheduled at the same time, it then should be cached in the same server. Constraints (14) and (15) ensure that the end-to-end delay between each cache server and the channels' subscribers should be lower than τ. Constraint (14) ensures that all scheduled videos Υ_i should be cached in a server where the end-to-end delay between that server and the subscribers of the current channel i is lower than τ. Similarly, constraint (15) ensures that the end-to-end delay between the subscribers of a channel i' and the chosen server l, for caching video j, is lower than τ. Constraints 16- 20 calculate, for each video j and server l, the number of streams $\eta_{j,l}$ that would be minimized by the objective function.

V. Simulation and Results

In this section, we evaluate the performance of the proposed solution using the Python language programing and the Gurobi optimization tool [25]. Table II shows the different metrics used for evaluating the performance of the proposed solution. To show the performance of our solution, we used a set of videos (100 videos) with durations varying between 10 and 80 minutes. These videos are randomly distributed on different channels, and a video can be in the program of different channels. Videos in the program of the same channel are scheduled while preventing the overlap of their playback times. The proposed solution is evaluated in terms of two metrics, namely 1) the number of streams generated in the network, formally defined as the total number of streams $\eta_{j,l}$ of all videos j; and 2) the runtime execution of the proposed solution.

TABLE II
Parameters used in the simulation.

Variable	Description
Number of channels	10
Number of videos	100
Number of servers	8
Number of regions	4
Video durations	10,20,40,80 min
Videos scheduled in one channel	1-10
Schedule duration	24 hours

Figure 3 shows the impact of the number of simultaneously scheduled videos on the proposed solution in terms of streaming cost (i.e., number of generated streams). The proposed solution is compared against a baseline solution, whereby no optimization is performed. In the baseline solution, each video is cached and streamed without taking into account if it is already scheduled in existing channels. As we have aforementioned, the QoS is always taken into account when selecting different caching servers, which creates more restriction on the proposed solution. Only the caching servers that have a low latency to the users should be selected. Although this restriction, we observe, from this figure, that the proposed solution outperforms the baseline solution, on average, by more than 50% in terms of number of streams. This means that when the proposed solution is used, the number of effective streams will be halved, which eventually reduces the overall cost. Moreover, the proposed solution outperforms the baseline solution by more than 60% when the number of simultaneously scheduled videos reaches 10.

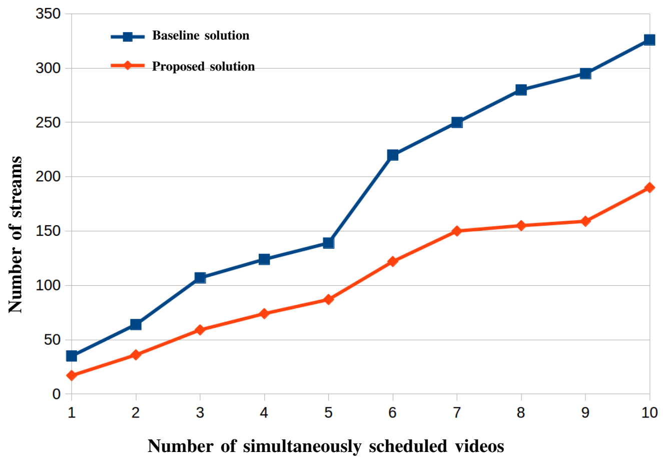

Fig. 3. The performance of the proposed solution in terms of reducing the streaming cost.

Figure 4 shows the impact of the number of simultaneously scheduled videos on the complexity of the proposed solution, measured by its executions. We varied the number of simultaneously scheduled videos between 1 and 10. The first observation that we draw from this figure that the execution time is reasonable. In all conducted simulations, the execution time does not exceed 0.30 sec. We also observe that the simultaneous scheduling of many videos requires less time if we schedule them separately, i.e., one by one. For example, scheduling one video requires 0.02955 sec while scheduling 8 videos requires 0.22511 sec (i.e., $\frac{0.22511}{8} = 0.028$ sec per video).

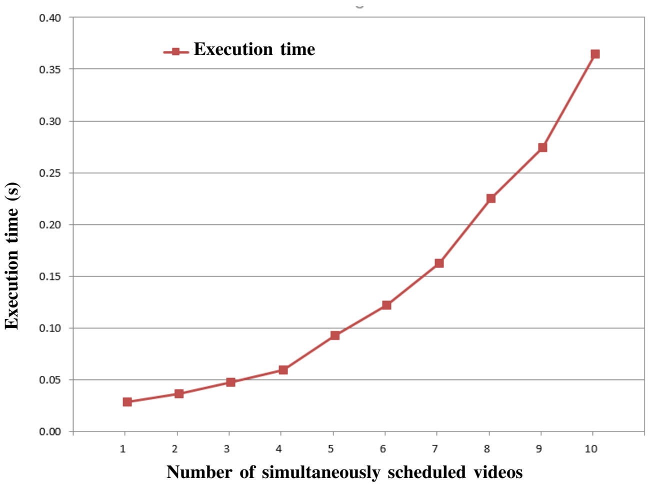

Fig. 4. The execution time of the proposed solution for varying number of simultaneously scheduled videos.

VI. Conclusion

In this paper, we proposed a solution for scheduling IPTV channels on top of the cloud. The proposed solution aims to minimize the cost while ensuring QoS. The cost in the proposed solution is minimized by managing the caching and streaming of videos in an efficient manner. The proposed solution schedules videos at the same time in the programs of as many channels as possible and places the different videos in the same cache servers, such that the number of effective streams created in the network is minimized. The obtained results demonstrate the efficiency of this strategy in reducing the caching and streaming cost while still maintaining QoS within an acceptable level.

References

[1] T. Taleb and K. Hashimoto. MS^2 : A New Real-Time Multi-Source Mobile-Streaming Architecture. *IEEE Transactions on Broadcasting*, 57(3):662–673, Sept 2011.

[2] J. C. Fernandez, T. Taleb, M. Guizani, and N. Kato. Bandwidth Aggregation-Aware Dynamic QoS Negotiation for Real-Time Video Streaming in Next-Generation Wireless Networks. *IEEE Transactions on Multimedia*, 11(6):1082–1093, Oct 2009.

[3] V. Aggarwal, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and V. A. Vaishampayan. Optimizing cloud resources for delivering iptv services through virtualization. *IEEE Transactions on Multimedia*, 15(4):789–801, June 2013.

[4] T. Taleb and N. Taleb. System and Method for Creating Multimedia Content Channel Customized for Social Network, Aug 2010. Patent number EP20110820657.

[5] S. C. Kim and S. K. Kim. Personalized IPTV content recommendation for social network group. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 469–470, Jan 2011.

[6] J. Yang, H. Park, G. M. Lee, and J. K. Choi. A web-based IPTV content syndication system for personalized content guide. *Journal of Communications and Networks*, 17(1):67–74, Feb 2015.

[7] Dong-Hee Shin. Defining sociability and social presence in social TV. *Computers in Human Behavior*, 29(3):939 – 947, 2013.

[8] Partha Mukherjee and Bernard J. Jansen. *Social TV and the Social Soundtrack: Significance of Second Screen Interaction during Television Viewing*, pages 317–324. Springer International Publishing, Cham, 2014.

[9] Frank Bentley, Karolina Buchner, and Joseph 'Jofish' Kaye. Mychannel: Exploring city-based multimedia news presentations on the living room tv. In *Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '14, pages 71–78, New York, NY, USA, 2014. ACM.

[10] P. A. Frangoudis, L. Yala, A. Ksentini, and T. Taleb. An architecture for on-demand service deployment over a telco CDN. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[11] S. Retal, M. Bagaa, T. Taleb, and H. Flinck. Content Delivery Network Slicing: QoE and Cost Awareness. In *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.

[12] S. Dutta, T. Taleb, P. A. Frangoudis, and A. Ksentini. On-the-Fly QoE-Aware Transcoding in the Mobile Edge. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016.

[13] F. Z. Yousaf and T. Taleb. Fine-grained resource-aware virtual network function management for 5g carrier cloud. *IEEE Network*, 30(2):110–115, March 2016.

[14] A. Ksentini and T. Taleb. QoE-Oriented Adaptive SVC Decoding in DVB-T2. *IEEE Transactions on Broadcasting*, 59(2):251–264, June 2013.

[15] A. Ksentini, T. Taleb, and K. B. Letaif. QoE-Based Flow Admission Control in Small Cell Networks. *IEEE Transactions on Wireless Communications*, 15(4):2474–2483, April 2016.

[16] D. O. Mau, T. Taleb, and M. Chen. MM3C: Multi-Source Mobile Streaming in Cache-Enabled Content-Centric Networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2015.

[17] Mau Dung Ong, Min Chen, Tarik Taleb, Xiaofei Wang, and Victor C.M. Leung. FGPC: Fine-grained Popularity-based Caching Design for Content Centric Networking. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '14, pages 295–302, New York, NY, USA, 2014. ACM.

[18] Sajal K Das, Zohar Naor, and Mayank Raj. Popularity-based caching for iptv services over p2p networks. *Peer-to-Peer Networking and Applications*, 10(1):156–169, 2017.

[19] T. Taleb. Toward carrier cloud: Potential, Challenges, and Solutions. *IEEE Wireless Communications*, 21(3):80–91, June 2014.

[20] Josep M. Pujol, Vijay Erramilli, Georgos Siganos, Xiaoyuan Yang, Nikos Laoutaris, Parminder Chhabra, and Pablo Rodriguez. The little engine(s) that could: Scaling online social networks. *SIGCOMM Comput. Commun. Rev.*, 40(4):375–386, August 2010.

[21] Xu Cheng and Jiangchuan Liu. Load-balanced migration of social media to content clouds. In *Proceedings of the 21st International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '11, pages 51–56, New York, NY, USA, 2011. ACM.

[22] Yu Wu, Chuan Wu, Bo Li, Linquan Zhang, Zongpeng Li, and Francis C. M. Lau. Scaling social media applications into geo-distributed clouds. *IEEE/ACM Trans. Netw.*, 23(3):689–702, June 2015.

[23] Feng Wang, Jiangchuan Liu, Minghua Chen, and Haiyang Wang. Migration towards cloud-assisted live media streaming. *IEEE/ACM Trans. Netw.*, 24(1):272–282, February 2016.

[24] T. Taleb, A. Ksentini, M. Chen, and R. Jantti. Coping With Emerging Mobile Social Media Applications Through Dynamic Service Function Chaining. *IEEE Transactions on Wireless Communications*, 15(4):2859–2871, April 2016.

[25] Gurobi Optimization - The Best Mathematical Programming Solver, url = http://www.gurobi.com/, urldate = 2016-10-28.