Optimizing Personalized Interaction Experience in Crowd-Interactive Livecast: A Cloud-Edge Approach

Haitian Pang[*,†], Cong Zhang[†], Fangxin Wang[†], Han Hu[§], Zhi Wang[‡], Jiangchuan Liu[†], Lifeng Sun[*]
[*]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[†]School of Computing Science, Simon Fraser University, BC, Canada
[§]Beijing Institute of Technology, Beijing, China
[‡]Graduate School at Shenzhen, Tsinghua University, China

ABSTRACT

Enabling users to interact with broadcasters and audience, the crowd-interactive livecast greatly improves viewer's quality of experience (QoE) and attracts millions of daily active users recently. In addition to striking the balance between resource utilization and viewers' QoE met in the traditional video streaming service, this novel service needs to take supererogatory efforts to improve the interaction QoE, which reflects the viewer interaction experience. To tackle this issue, we conduct measurement studies over a large-scale dataset crawled from a representative livecast service provider. We observe that the individual's interaction pattern is quite heterogeneous: only 10% viewers proactively participate in the interaction, and the rest viewers usually watch passively. Incorporating the insight into the emerging cloud-edge architecture, we propose a framework PIECE, which optimizes the Personalized Interaction Experience with Cloud-Edge architecture (PIECE) for intelligent user access control and livecast distribution. In particular, we first devise a novel deep neural network based algorithm to predict users' interaction intensity using the historical viewer pattern. We then design an algorithm to maximize the individual's QoE, by strategically matching viewer sessions and transcoding-delivery paths over cloud-edge infrastructure. Finally, we use trace-driven experiments to verify the effectiveness of PIECE. Our results show that our prediction algorithm outperforms the state-of-the-art algorithms with a much smaller mean absolute error (40% reduction). Furthermore, in comparison with the cloud-based video delivery strategy, the proposed framework can simultaneously improve the average viewers QoE (26% improvement) and interaction QoE (21% improvement), while maintaining a high streaming bitrate.

CCS CONCEPTS

• **Networks** → **Cloud computing**; *Overlay and other logical network structures*; • **Computer systems organization** → **Cloud computing**; *Neural networks*; *Real-time system architecture*;

KEYWORDS

Interactive Live Streaming, Cloud-Edge, Viewer Interaction

ACM Reference Format:
Haitian Pang, Cong Zhang, Fangxin Wang, Han Hu, Zhi Wang, Jiangchuan Liu, Lifeng Sun. 2018. Optimizing Personalized Interaction Experience in Crowd-Interactive Livecast: A Cloud-Edge Approach. In *2018 ACM Multimedia Conference (MM '18), October 22–26, 2018, Seoul, Republic of Korea.* , 9 pages. https://doi.org/10.1145/3240508.3240642

MM '18, October 22–26, 2018, Seoul, Republic of Korea

ACM ISBN 978-1-4503-5665-7/18/10…$15.00
https://doi.org/10.1145/3240508.3240642

1 INTRODUCTION

Nowadays, the crowd-interactive livecast (CIL) has been the main fuel of the rapid growth of live streaming, such representative services as YouTube Gaming[1], Douyu.tv[2], Inke.tv[3], and Twitch.tv[4] have attracted millions of daily active users. The salient feature of CIL is allowing common users (broadcasters) to broadcast live streaming to viewers using the ordinary devices, and enabling viewers to interact with the broadcaster and other viewers. Except for the streaming transcoding and delivery modules, a representative CIL service also includes an interaction module enabling viewers to send interactive messages, and deliver the interaction messages to the broadcaster as well as the viewers.

Although previous interactive streaming services allow viewers to perform operations such as fast forward and pause, never before has the crowd interaction been so important like in today's CIL service. Viewers in the CIL service not only watch the streaming itself, but also participate in the interaction. As a practical example, a significant amount of 14.2 billion interactive messages are sent in the Twitch platform in 2016[5].In the TwitchPlaysPokemon[6] channel, the game progress is even determined by the majority of the viewers' opinion. In summary, viewers are highly engaged to the livecast with the frequent interactions and their behaviors also noticeably affect the livecast process. As a result, conventional streaming QoE definition [17] considering the bitrate, the startup delay, and rebuffer cannot characterize the interaction experience of viewers. Instead, the viewer QoE consists of not only the streaming QoE, but also the interaction QoE. Specifically, the interaction QoE is affected by whether the streaming and the interactions are synchronous, and whether sending an interaction can be immediately responded by the broadcaster. Once a viewer with a large latency sends an

[1]https://gaming.youtube.com/
[2]https://www.douyu.com/
[3]https://www.inke.com/
[4]https://www.twitch.com/
[5]https://www.twitch.tv/year/2016/
[6]https://www.twitch.tv/twitchplayspokemon

interaction message, all the other viewers and the broadcaster will see the interaction with a large latency, harming the interaction experience of all the parties.

To study the viewers' interaction pattern, we conduct measurement on a large-scale interaction dataset collected by crawling the Twitch.tv's API. From the crowd's perspective, the interaction is massive. However, from the individual's perspective, the interaction is quite heterogeneous. We define the viewers who frequently interact with the broadcaster and other viewers as *proactive* viewers, and the viewers seldom interact with others as *passive* viewers. With the above definition, we observe the following viewer patterns: 1) Viewers are heterogeneous in terms of interaction frequency, i.e., 10% *proactive* viewers generate a large portion of interactive messages, while the rest 90% *passive* viewers seldom interact. 2) The crowd viewers' interaction intensities in different broadcast channels are diverse. 3) One viewer has diverse interaction behaviors in different broadcast channels. With the awareness of the viewer's interaction pattern, we can conclude that the *proactive* viewers are more sensitive to large latency compared to the *passive* viewers. For example, *proactive* viewers cannot tolerate large latency, as the messages they send to the broadcaster are severely delayed. If a *proactive* viewer bears large latency, all the others' interaction QoE will be harmed due to delayed interaction. However, *passive* viewers can bear large latency, as long as the streaming and the interaction are synchronous.

Besides the heterogeneous requirement for the latency brought up by personalized interaction pattern, the requirement for the bandwidth is also overwhelming in the CIL service as the broadcast channels are numerous, not to mention the viewers. Thus, an ideal scheme to satisfy the personalized QoE requirement of each viewer is to choose the network path with the proper metrics. Many architectures have been proposed for live streaming delivery [5], including the centralized architecture, e.g., CDN [8] and cloud-based approaches [1], and the distributed architecture, e.g., P2P [6] and crowdsourced [2] approaches. However, the centralized architecture fails to provide fine-grained service as the server number is limited, while the distributed architecture will introduce instability and latency overhead. Satisfying each individual viewer's personalized QoE requirement is challenging, as this calls for ultra-flexible and fine-grained management of the transcoding and delivery architecture in the granularity of each viewer.

To tackle the above problems, we develop a framework PIECE to optimize the personalized interaction experience with cloud-edge architecture. Specifically, we design a novel framework to predict viewer interaction patterns and utilize the results to lead the streaming transcoding and delivery in the newly emerged cloud-edge architecture. The new design philosophy allows us to optimize the personalized viewer's QoE with the awareness of viewers' interaction pattern. As the viewer's interaction pattern is quite complicated and the data volume of the interaction is large, the challenge arises as to how to predict the viewer interaction precisely. Considering that the traditional machine learning methods may not perform well on the large data volume, we design a powerful deep neural network framework to predict the viewer interaction pattern, which performs better with more training data.

Recently, the edge network has emerged as a novel approach to deliver the video streaming [3, 13], which can reduce the delivery latency and the burden of the cloud. Yan *et al.* [15] implemented a novel system to integrate CDNs and edge clouds. Pang *et al.* [9, 10] utilized the edge network to improve the broadcaster uploading performance in crowdsourced live streaming. Ma *et al.* [7] used an edge computing-assisted paradigm to deliver the streaming in mobile personal livecast. Despite the interest from the academia, edge computing also attracts the attention of the industry. NVIDIA released an edge computing device in March, i.e., Jetson TX2, which supports 4K x 2K 60 Hz video encoding[7]. In the cloud-edge architecture, the cloud serves as the centralized component guaranteeing sustainable service quality and the edge devices serve as the distributed components to provide diverse service quality for personalized viewers. Specifically, each viewer can achieve different network performance by choosing different edge devices. For example, choosing a low-latency path for the *proactive* viewers, and choosing a high-bitrate path for the *passive* viewers, via matching the viewer session with the proper edge device. In summary, the most important benefit of the cloud-edge architecture is flexible and fine-grained network management for each viewer. The cloud-edge architecture also has the following benefits: (1) The transcoding and bandwidth load on the cloud can be decreased dramatically with the assistance of the edge. (2) As the edge devices are closer to viewers, the startup delay can also be decreased by assigning the viewers to nearby edge devices.

To the best of our knowledge, our study is the first to explore the interaction pattern and utilize the cloud-edge architecture in crowd-interactive livecast. Our contributions are summarized as follows: (1) We conduct large-scale measurement studies to get insight into the viewer interaction pattern in the CIL service. (2) We propose the PIECE framework to optimize the CIL service. (3) We design a deep neural network framework to accurately predict the viewer interaction pattern. (4) With the assistance of the cloud-edge architecture and the viewer's interaction prediction, we formulate the viewers' personalized QoE optimization problem and design a practical algorithm to solve it.

The rest of the paper is organized as follows: We provide the measurement results in Section 2, and provide the system overview of the PIECE framework accordingly in Section 3. We further provide the interaction prediction model in Section 4. The QoE representation is illustrated in in Section 5. The viewers' personalized QoE optimization problem in the cloud-edge architecture is presented in Section 6. We provide the experiment results in Section 7 and conclude in Section 8.

2 MEASUREMENT

In this section, we conduct large-scale measurement studies on viewer's interaction pattern.

2.1 Dataset Description

We collected the interactive messages from Twitch.tv, which is the pioneer ILS in the world with 15 million daily active users (DAU) by February 2018[8]. In the rest of the paper, we use interaction and message interchangeably. The interaction data of Twitch was collected by an Internet Relay Chat Protocol based crawler in

[7] https://developer.nvidia.com/embedded/develop/hardware
[8] http://twitchadvertising.tv/audience/

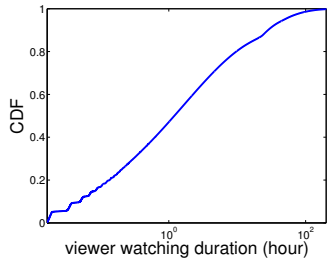

Figure 1: CDF for the total watching duration of each viewer.

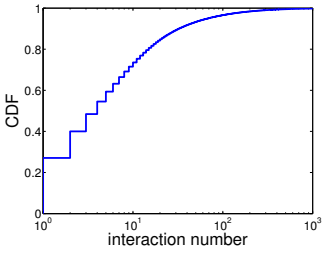

Figure 2: CDF for the total interactions of each viewer.

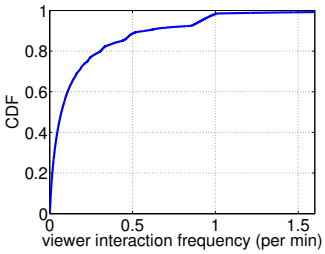

Figure 3: CDF of viewers' interaction frequency.

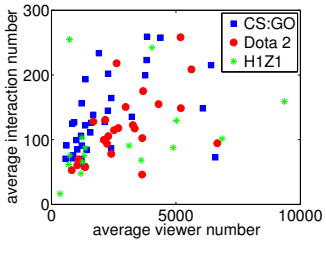

Figure 4: An example distribution of channels in average viewer and interaction number.

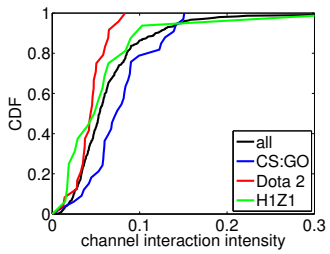

Figure 5: CDF of the channel interaction intensity.

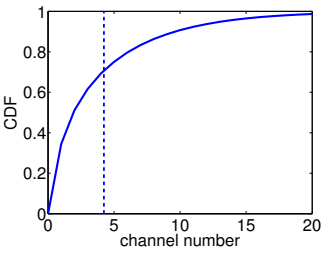

Figure 6: The CDF of visited channel number for each viewer.

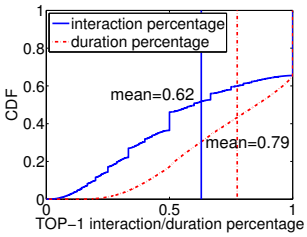

Figure 7: CDF of interaction and duration percentage for the TOP-1 channel.

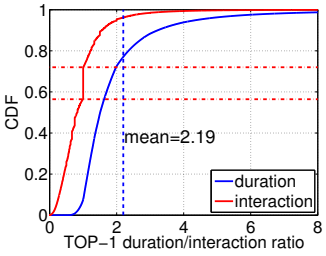

Figure 8: CDF of the duration/interaction ratio for the TOP-1 channel.

50 consecutive days. Using the crawler, we can join multiple channels and gather the messages and viewer data from 300 most popular Twitch channels[9]. The items we can derive are "viewer id", "broadcaster id", "time", and "action". There are three types of actions, we capture "JOIN" when a viewer enters a channel, "PART" when a viewer exits a channel, and "MSG" when a viewer sends a message. We can further extract the content from the message.

During this time period, we collected 158.3 million viewer messages in the 300 channels. We also collected all the viewer data in the channels, which contains 6.7 million unique viewers establishing 99.3 million viewer sessions. The channels can be classified into 45 categories by the content type, most of which are games.

2.2 Measurement Results

2.2.1 Viewer Patterns. We first investigate the total watching duration and the total interaction number of all viewers. We show the cumulative distribution function (CDF) of total watching duration and interaction number of viewers in Figure 1 and Figure 2, respectively. We observe that both the watching duration and the interaction number is highly skewed. The distributions indicate that there are a dominant fraction of passive viewers in the crowd-interactive livecast.

We define the viewer interaction frequency as the average interaction number generated by the viewer per unit time (1 minute), calculated as $\frac{\#\text{ of interaction}}{\#\text{ of duration}}$. The interaction frequency illustrates the viewer activeness when watching live streaming. Figure 3 shows the CDF of the interaction frequency. We notice that the distribution of the interaction frequency is highly skewed, and over 90% viewers' interaction frequency is lower than 0.5.

Figure 1 to Figure 3 indicate that a small portion of proactive viewers post most of the messages, while most of the passive viewers watch the livecast but send few messages. The prevalent methods ignore the interaction diversity and may cause QoE degradation for these proactive viewers and the corresponding broadcasters.

2.2.2 Channel Patterns. We can calculate the average interaction number of one channel by averaging all the interaction number within the unit time. Similarly, we also calculate the average viewer number of one channel by averaging all the online viewer number within the unit time. We plot the relationship between the average interaction and viewer number in Figure 4 of three games, i.e., CS:GO, Dota 2, and H1Z1. We choose the three games which is representative game of FPS[10], MOBA[11], and BRG[12].

We notice that channels are quite diverse as to the two metrics. Although the CS:GO channels have few viewers, the viewer interaction is proactive. The H1Z1 channels have many viewers, but the interaction is not proactive. This may result from the broadcaster and the game type. Thus, we need to better characterize the channels. Next, we define the interaction intensity for each channel, which is computed as $\frac{\#\text{ of average interaction}}{\#\text{ of average viewer}}$. We show the interaction intensity distribution in Figure 5, which is highly skewed. This also validates that viewers send more messages when watching CS:GO.

The above observations show that the interaction intensity is heterogeneous across channels, and the channel type affects the interaction intensity obviously. The prevalent methods ignore the interaction diverse across channels and may incur QoE degradation for certain types of channels.

2.2.3 Viewer-Channel Pair Patterns. We further investigate how many channels the viewer visited during the time period. Figure 6 illustrates the channel number distribution of viewers. We notice

[9]https://help.twitch.tv/customer/portal/articles/1302780-twitch-irc

[10]https://en.wikipedia.org/wiki/First-person_shooter
[11]https://en.wikipedia.org/wiki/Multiplayer_online_battle_arena
[12]https://en.wikipedia.org/wiki/Battle_royale_game

that the average channel number that the viewers visited is 4.23, and 90% viewers watch no more than 10 channels. Another observation is that 33% of viewers visit only one channel.

As the number of visited channels is relatively small, we further focus on the TOP-1 channel for every viewer. The TOP-1 channel is defined as the channel in which a viewer watch the longest duration. We first show the percentages of the watching duration and the interaction number generated from the TOP-1 channel as depicted in Figure 7. The first observation is that 33% viewers visit only one channel. In this case, the relative duration and interaction equal 1. The second observation is that the average interaction and duration percentages are relatively high: 62% of interactions are generated from the TOP-1 channel, and 79% of the duration is spent in the TOP-1 channel. These observations show that most viewers are attracted by one channel, which further motivates us to investigate the viewer pattern in the TOP-1 channel.

Next, we define the duration ratio as the average duration in the TOP-1 channel divided by the average duration in all channels. If the duration ratio is larger than 1, the viewer will stay longer in the TOP-1 channel. Similarly, the interaction ratio is defined as the average interaction in the TOP-1 channel divided by the average interaction in all channels. If the relative interaction is larger than 1, the viewer will send more messages in the TOP-1 channel. As the relative duration and interaction are constant 1 for viewers visiting only one channel, the analysis is especially for the viewers visiting more than one channel.

We provide the distribution of the duration and interaction ratio in Figure 8. We observe that 91% of viewers will stay longer in the TOP-1 channel, and the average duration ratio is 2.19, indicating that viewers will stay in the TOP-1 channel for more than 2 times longer. Counterintuitively, the interaction number does not show the similar pattern as the duration distribution. Only 28% of viewers interact more in the TOP-1 channel, and 56.4% of viewers interact fewer in the TOP-1 channel. This may be caused by the diverse viewer behaviors: some viewers watch the TOP-1 channel passively, i.e., they focus on the streaming itself and the interaction between the broadcaster and other viewers. Some viewers watch the TOP-1 channel actively, i.e., they prefer to communicate and influence the streaming process, as they are familiar with the broadcaster.

From the above analysis, we observe that viewers will stay longer in the TOP-1 channel, while the interaction frequency is determined by whether a viewer is proactive or passive. The prevalent methods ignore the interaction diverse across the viewer-channel pairs, and may incur QoE degradation.

3 PIECE: A SYSTEM OVERVIEW

Motivated by the aforementioned measurement insights, we introduce the cloud-edge based system design for crowd-interactive livecast, and detail the workflow of our system.

Figure 9 shows the cloud-edge architecture specifically designed for the CIL service. The cloud-edge architecture can offload the transcoding and delivery cost to edge devices, provide service with lower latency via the close-to-user edge devices, and most importantly enable flexible and fine-grained network management. In our scenario, broadcasters upload the raw streaming to the cloud. Then, the cloud transcodes the streaming to only high resolutions,

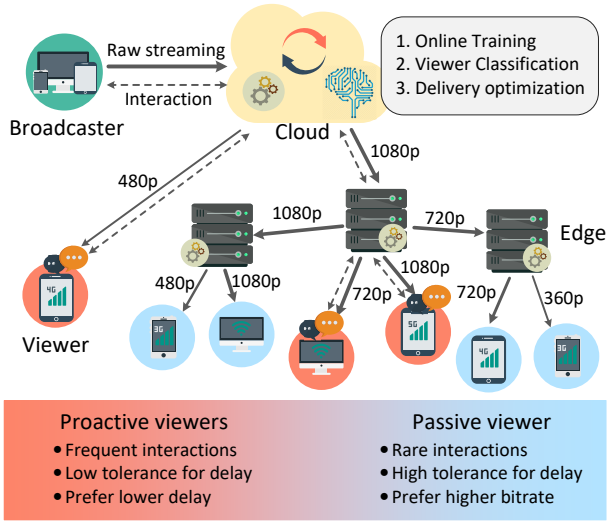

Figure 9: Design of the PIECE framework.

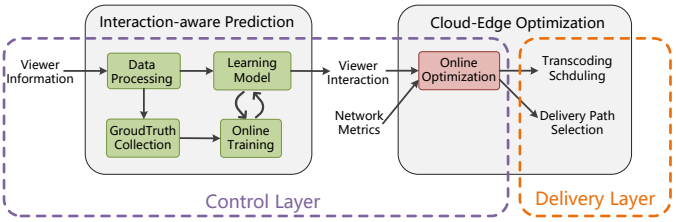

Figure 10: System Workflow.

and deliver to edge devices. Then, edge devices transcode the video to the requested bitrates and deliver to viewers. As the viewers have different interaction patterns, the proactive viewers can be assigned to a low-latency path, and the passive viewers can be assigned to a high bandwidth path. The challenge is to predict the viewer interaction pattern and choose a proper edge device for her.

Figure 10 shows the workflow of the system. The system can be divided into two layers by the function, i.e., the control layer and the delivery layer. In the control layer, the cloud collects the viewers' historical data and perform proper preprocessing. On one hand, the real-time request data is fed to the learning model to predict the viewer's interaction behavior. On the other hand, when the ground truth is collected, the model training process is triggered to update the model online. Besides, the network metrics are also collected for intelligent decision-making. With the predicted interaction behavior and the measured network metrics, an intelligent decision is made to optimize the viewer's delivery path via performing the transcoding and delivery in the control layer. Next, we will introduce the interaction prediction module and the cloud-edge delivery module, respectively.

4 INTERACTION PREDICTION MODEL

In this section, we first introduce how to preprocess the input data to get an integrated input vector for the neural network. Then, as the interaction number is highly imbalanced as shown in the measurement, we design a cost-sensitive neural network (CSNN) to cope with the problem.

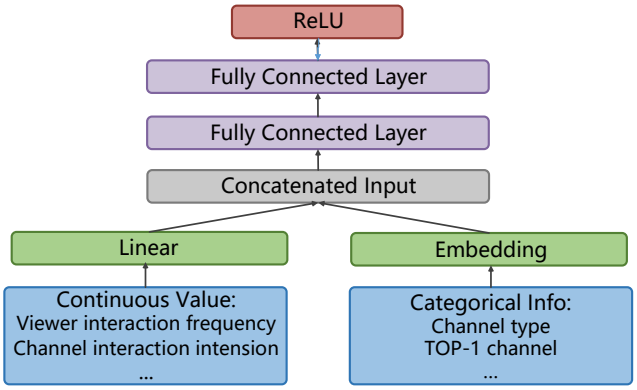

Figure 11: The nerual network architecture.

4.1 Data Preprocessing

With the insights provided in the measurement section, we design a deep neural network model to predict the interaction frequency. However, the dataset heterogeneity poses great challenge for efficient learning in the following aspects: (1) some data fields describe the categorical information (content type), while some others are numerical values (viewer's average duration); (2) finding a proper representation for the categorical information can be difficult, as the category number may be large, and there exists inherent relationship among the categories.

The input data includes viewer's average watching duration, average interaction number, and interaction frequency, channel's category and interaction intensity, and viewer's average interaction frequency, duration and duration rank in the channel. To address the above challenges, we divide the input data into two types: categorical information, and numerical values. For the numerical values, we simply use a linear preprocessing layer to process the input. For the categorical information, as the conventional one-hot encoding is high-dimensional and sparse, the input vector grows with the category number. This results in computational inefficiency and low accuracy. For this reason, we need to find low-dimensional and dense representation methods for the categorical information. Thus, we design an embedding layer for the categorical information to reduce the dimension. Then, we concatenate the preprocessed input and feed them to the neural network, as shown in Figure 11.

4.2 Cost-Sensitive Neural Networks

The neural network is built by stacking multiple fully connected layers, and the output layer is one neuron returning the predicted interaction number when a session is established. As the ground truth interaction number ranges from zero to infinity, we choose the ReLU function as the activation function for the output neuron in the network, whose output is in the range $[0, +\infty)$. In addition, the activation function in the fully connected layers is set as *tanh*. In the training phase, the neural network used for regression is usually trained to minimize the mean square error.

5 QOE REPRESENTATION IN CIL SERVICE

This section provides the QoE representation in the CIL system. We first formulate the viewer's QoE, and then design the conceptual viewer's interaction QoE.

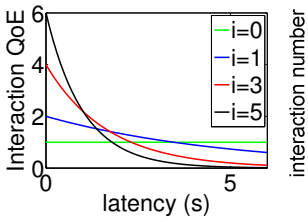

Figure 12: The example of the interaction function. The parameter settings are $a = 1.5$, and $b = 0.2$.

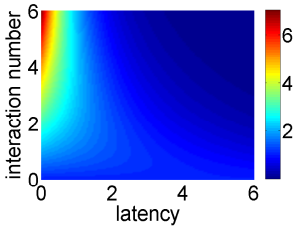

Figure 13: The values of the interaction QoE under different latency and interaction number settings.

5.1 Introducing the Viewer's QoE

In our design, taking advantage of the edge devices, a viewer can be directed to an ideal path. This design principle allows viewers to choose the edge device with the largest QoE based on her own interaction pattern. We formulate this problem as an optimization problem. We denote the viewer set as $\mathbb{V} = \{1, 2, ..., V\}$.When a viewer watches a broadcast channel, the corresponding viewer-channel session is formed. Each session bears an interaction index i_v, which is the estimated viewer interaction number during the session. Recall that the value of i_v is determined by viewer v, the broadcast channel b, and other context information. We use the CSNN method introduced in Section 5 to predict the value of i_v. We further denote the bitrate and latency as b_v and l_v, respectively. Consequently, we can formulate the viewer's QoE as follows:

$$Q_v = \alpha \cdot b_v + \beta \cdot I(l_v, i_v) - \gamma \cdot l_v^+, \tag{1}$$

where the first term reflects the bitrate, which is widely adopted in the QoE definition of video streaming. The second term is the interaction QoE, a component we design to evaluate the viewer's interaction experience under certain setup of l_v and i_v. The third term is the startup delay, which is the delay between the initiated streaming source and the viewer. Note that the startup delay l_v^+ is no longer than the broadcaster-viewer delay l_v by nature. α, β and γ are the weight parameters.

5.2 Formulation of the Interaction QoE

Next, we provide the detailed definition of the interaction QoE. The interaction QoE is affected by the latency and the interaction number following the three rules: (1) a lower latency will reduce the communication uncomfortableness between the broadcaster and the viewers, thus incurring higher interaction QoE. (2) viewers sending more interaction messages expect a lower latency more than viewers that seldom send interaction messages. (3) If the latency equals zero, i.e., the broadcaster and the viewers can communicate in real-time, viewers sending more interaction message will achieve higher interaction QoE.

Inspired by [12], we choose the exponential function e^{-x} to characterize the interaction QoE. This function captures the notion that the marginal reduction in QoE decreases at longer latency. We provide the example function as follows:

$$I(i_v, l_v) = (a + i_v) \cdot e^{-b \cdot i_v l_v}, a, b > 0, \tag{2}$$

where a and b are the weight parameters. The mathematical properties of the above QoE function are as follows. When we fix the interaction number arbitrarily as i_v^*, $I(l_v, i_v^*)$ is a decreasing

function of l_v to reflect the viewer latency, which can adjust the interaction QoE according to l_v so that a larger latency will induce lower viewer QoE. Formally, this is represented as follows:

$$I(l_v^1, i_v^*) > I(l_v^2, i_v^*), \quad \forall l_v^1 < l_v^2,\ i_v^* \tag{3}$$

Different values of i_v affect the derivative of $I(l_v, i_v^*)$ for l_v. A larger i_v^* corresponds to faster decrease of $I(l_v, i_v^*)$. Formally, this is represented as follows:

$$I(l_v^1, i_v^1) - I(l_v^2, i_v^1) < I(l_v^1, i_v^2) - I(l_v^2, i_v^2),\ \forall l_v^1 < l_v^2,\ i_v^1 < i_v^2. \tag{4}$$

When $l_v = 0$, proactive viewers will achieve higher interaction experience, as they can interact with the broadcasters without any delay. Formally, this is represented as follows:

$$I(0, i_v^1) > I(0, i_v^2), \quad \forall i_v^1 > i_v^2, \tag{5}$$

We show the example interaction QoE functions in Figure 12, where $i_v = 0, 1, 3, 5$. We further provide the values of the interaction QoE under different latency and interaction number settings in Figure 13, which validates the above three rules.

6 INTERACTION-AWARE OPTIMIZATION

This section provides the network optimization of the cloud-edge architecture, which is built on the viewer interaction pattern.

After introducing the viewer's QoE function, our aim is to choose the proper path for the viewer to maximize the QoE. We denote the edge device set as $\mathbb{E} = \{1, 2, ..., E\}$, and the regional server set as $\mathbb{S} = \{1, 2, ..., S\}$. Viewers can use different edge devices and cloud servers as delivery path options, incurring different network latencies and bitrates. That is to say, l_v and b_v are the functions of the edge devices and the servers. The edge device and the server selected for viewer v is denoted as e_v and s_v. Hence, the viewer's QoE can be formulated as the function of the chosen edge device e_v and server s_v:

$$Q_v(e_v, s_v) = \alpha \cdot b_v(e_v, s_v) + \beta \cdot I(l_v(e_v, s_v), i_v) - \gamma \cdot l_v^+(e_v, s_v). \tag{6}$$

The latency of choosing a path can be calculated by adding the latency between the viewer and the edge device and the latency between the edge device and the server. The end-to-end latency measurement methods can be derived using previous work [4]. The startup delay can be measured in a similar way. Once there exist the requested or higher representations, the streaming can be delivered from the edge device directly, thus the startup delay is the latency between the edge device and the viewer. Otherwise, the streaming has to be delivered from the server, thus the startup delay is the latency between the server and the viewer.

As the viewer interaction patterns are quite heterogeneous in terms of viewer, channel, and viewer-channel pair, each viewer's optimal decision depends on not only the network performance but also the viewer interaction pattern. Viewer v's optimal selection is:

$$(e_v^*, s_v^*) = \arg\max_{(e,s)} Q_v(e, s),\ \forall e \in \mathbb{E}, s \in \mathbb{S}. \tag{7}$$

The bitrate can be calculated by finding the bandwidth bottleneck from the viewer-to-edge bandwidth B_{ve}, and the edge-to-server bandwidth B_{es}. The server delivers the unique viewer requests in terms of the channel and the bitrate to edge device e. B_{ve} can be directly measured, and B_{es} can be calculated by the total bandwidth B_{es}^* minus all the representations via edge device e. If the current

Algorithm 1 Interaction-aware Cloud-Edge Selection Algorithm

```
procedure EDGE DEVICE SELECTION(v, b)
    B_es = B*_es, ∀e ∈ 𝔼, s ∈ 𝕊
    if viewer v arrives: then
        Predict i_v with the deep learning model
        (e*_v, s*_v) = arg max Q_v(e, s), ∀e ∈ 𝔼, s ∈ 𝕊
        if no representation or lower representation then
            Request streaming from the server
            B_{e*_v s*_v} ← B_{e*_v s*_v} − b_v(e*_v, s*_v)
        end if
    end if
    if viewer v departures: then
        if no b_v(e*_v, s*_v) or lower resolution request then
            B_{e*_v s*_v} ← B_{e*_v s*_v} + b_v(e*_v, s*_v)
        end if
    end if
end procedure
```

bitrate representation of the channel on the edge device is larger than B_{ve}, it will either be delivered to the viewer directly or be transcoded to the expected representation by the edge device and then delivered to the viewer. We denote the available bitrate set as $br = \{br_1, br_2, ..., br_n\}$. Thus, the bitrate is calculated as follows:

$$b_v(e_v, s_v) = \arg\max_{b_v \in br} b_v,\ \forall b_v < \min B_{ve}. \tag{8}$$

If no representation exists on the edge device, the allowed bitrate is determined by the bottleneck of both the B_{ve} and B_{es}. The bitrate is calculated as follows:

$$b_v(e_v, s_v) = \arg\max_{b_v \in br} b_v,\ \forall b_v < \min(B_{ve}, B_{es}). \tag{9}$$

If a lower representation b^+ exists on the edge device, the bitrate is calculated as follows:

$$b_v(e_v, s_v) = \arg\max_{b_v \in br} b_v,\ \forall b_v < \min(B_{ve}, B_{es} + b^+), \tag{10}$$

in which case, the server will send the $b_v(e_v)$ bitrate representation to replace the b^+ representation.

The interaction-aware cloud-edge selection algorithm is illustrated in Algorithm 1. The algorithm works when a viewer enters a channel: predicting the viewer interaction and perform the path selection. When a viewer exits a channel, the network resource is released. The complexity comes from the traverse of the edge-server pair, so the operation time is $O(ES)$.

7 EXPERIMENT RESULTS

We first provide the experiment setup, and then evaluate the proposed CSNN interaction prediction algorithm. The performance of PIECE is provided in the end.

7.1 Experiment Setup

We conduct both the viewer's interaction prediction and the QoE optimization experiment on the top 300 channels in our dataset. We assume that 3 regional cloud servers are deployed for video delivery. The default number of geo-distributed edge devices is set to 80, and the available bandwidth of each edge device is set as 150Mbps. As

Table 1: Measurement results on Twitch of bandwidth requirement and transcoding cost.

resolution	1080p60f	1080p	720p60f	720p	480p	360p
bandwidth (Mbps)	5.86	4.45	2.75	1.93	1.1	0.52
transcoding (vCPU usage)	454.46%	332.91%	210.28%	141.71%	81.6%	50.51%

Table 2: The RMSE and MAE of different interaction prediction algorithms.

algorithm	Decision Tree	Random Forest	Naive Bayes	CSNN
RMSE	14.332	7.696	10.652	6.674
MAE	2.529	1.865	1.941	1.106
training time (s)	9.44	13.02	2.88	215.89

shown in Table 1, we measure the average bandwidth requirements for different resolutions by Twitch's official video statistic tool. We further evaluate the transcoding cost of each resolution on the AWS c4.8xlarge instance[13], which is allocated 36 vCPUs [14], as shown in Table 1. The weight parameters of the QoE are set as follows: $\alpha = 2$, $\beta = 3$, and $\gamma = 1$, which normalize the three terms to close values. The parameters defined in the Interaction QoE function are set as follows: $a = 1.5$, $b = 0.2$.

7.2 Performance of the Interaction Prediction

The number of the fully connected layer is set to 5, and the neuron number in each layer is set as 16 in CSNN. The category number is embedded as 10 dimensions. With the features extracted in Section 2, we compare the CSNN algorithm with the conventional regression algorithms: Decision Tree, Random Forest, and Naive Bayes. Note that we use the above regression algorithms in the *scikit-learn* tool kit [11]. We divide the whole dataset into the train set (the first 30 days) and the test set (the last 20 days) by the temporal sequence. Since the problem is cast as a regression problem, the ground truth is the true interaction number. To evaluate the performance of the CSNN algorithm, we calculate the mean absolute errors (MAE) and root mean square errors (RMSE).

Table 2 shows the comparison results on the Twitch dataset at the aspect of the regression error and the execution time. Both RMSE and MAE reflect the prediction error, and our algorithm achieves the lowest values among all the algorithms. Specifically, the proposed CSNN algorithm can obtain 40% lower MAE than the best result of the random forest algorithm. Despite achieving lower MAE, our algorithm also obtains 13% lower RMSE than the best result of the random forest algorithm. The reason that our algorithm performs best is that we efficiently solve the input heterogeneity and the data imbalance problems. In addition to the prediction error, we also investigate the execution time of each algorithm. The training time of different algorithms range from several seconds to several minutes, which is negligible considering that the time span of the training data is one month. We notice that the training time of CSNN is the longest, indicating CSNN sacrifices computation to reduce the prediction error.

7.3 Evaluation of the PIECE Framework

Next, we evaluate the performance of PIECE. We compare PIECE with (1) the traditional cloud architecture with non-interaction [18] (C-nI): The path selection relies on the cloud, and the viewer's interaction is not considered. Thus, the objective QoE function of this method is [16]:

$$Q_v = \alpha \cdot b_v - \gamma \cdot l_v^+. \tag{11}$$

(2) interaction-aware cloud architecture (CI): The path selection relies on the cloud, considering the viewer's interaction. We further use CSNN and random forest (RF) as the interaction prediction algorithms, namely CI-CSNN, CI-RF. (3) cloud-edge non-interaction architecture (CE-nI): The path selection relies on cloud and edge, without considering the viewer's interaction. (4) interaction-aware cloud-edge architecture with random forest (CEI-RF).

We first evaluate the viewer's average QoE under different edge device numbers, as depicted in Figure 14. We derive the following observations: (1) As the C-nI, CI-RF, and CI-CSNN methods do not use the edge devices for distribution and transcoding, the achieved QoE is identical to different number of edge devices. (2) The average QoE of the CE-nI, CEI-RF, and PIECE methods increase with the number of edge devices, as deploying more edge devices provides more options for delivery path selection. (3) Our proposed PIECE method outperforms the baselines including the CEI-RF method under all the edge device numbers. For example, when the number of the edge device reaches 32, our method increase the average viewer's QoE by 26% compared to CI, and outperform the traditional cloud-based method by 67%. (4) Using CSNN as the prediction algorithm outperforms the RF method by 10% as CSNN has lower prediction error. (5) Interaction-aware methods outperform the corresponding methods which do not consider viewer's interaction by optimizing the newly proposed interaction QoE.

Moreover, we investigate the viewer's average QoE under different capacities of the edge device, which is shown in Figure 15. The average viewer's QoE of the cloud-edge methods increase with the capacity (available bandwidth) of the edge device. However, we observe that the average viewer's QoE is identical when the capacity is in range $[0, 5]$. The reason is that the bitrate bears large weight in the conventional QoE definition, while ignoring the viewer's interaction experience. Thus, the edge devices with low capacity are not utilized even if they may achieve better interaction experience. When the capacity reaches 32, PIECE outperforms the baselines by 16%.

To better understand the QoE achieved by PIECE, we analyze the individual components in our overall QoE definition. Specifically, Figure 16 compares PIECE with the baselines in terms of bitrate, interaction experience, and the startup delay. Specifically, PIECE does not outperform the baselines on all the QoE components. Instead, it can balance the components to optimize the QoE: PIECE focuses on achieving the higest interaction QoE and the lowest startup delay, while retaining sustainable bitrate. Specifically, our method can improve the interaction QoE by 21% than the best

[13] https://aws.amazon.com/ec2/pricing/on-demand/

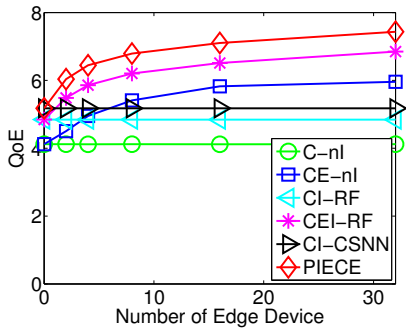

Figure 14: Average viewer QoE versus the number of edge devices.

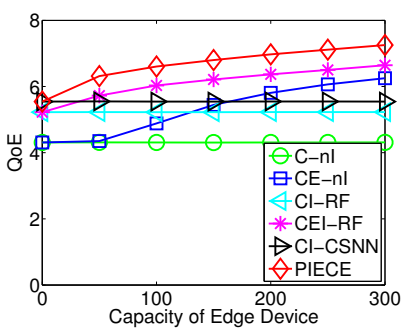

Figure 15: Average viewer QoE versus the capacities of the edge device.

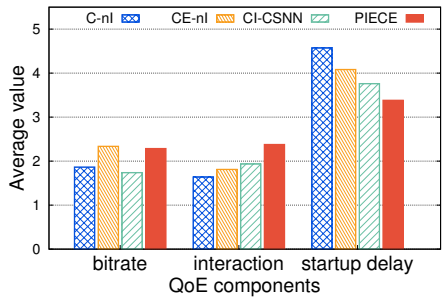

Figure 16: Comparing ECI with baselines by analyzing the performance on the individual components in the QoE definition.

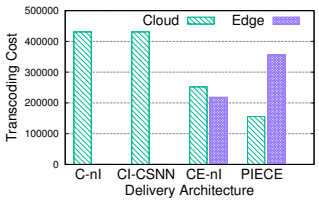

Figure 17: Transcoding cost in cloud and edge.

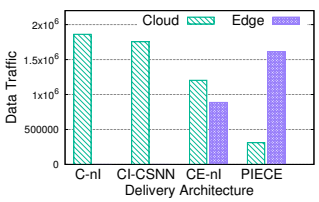

Figure 18: Bandwidth consumption in cloud and edge.

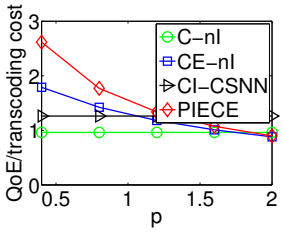

Figure 19: $\frac{\text{QoE}}{\text{transcoding cost}}$ versus transcoding price ratio p.

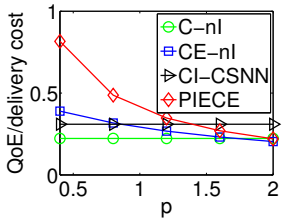

Figure 20: $\frac{\text{QoE}}{\text{delivery cost}}$ versus delivery price ratio p.

baseline, i.e., CI-CSNN, and can lower the startup delay by 30% than the wide-adopted centralized content delivery.

We further investigate the transcoding cost and the data traffic consumed in the cloud and the edge. Figure 17 shows the transcoding cost comparison of the proposed method and the baselines. Recall that the transcoding computing cost is measured in Table 1. We notice that the transcoding cost in the cloud is identical for C-nI and CI-CSNN, as the source streaming is transcoded to all possible resolutions in the cloud. The proposed PIECE method explores more on the edge devices than CE-nI, indicating that the interaction experience can be better satisfied with the edge devices. However, the cloud-edge architecture may induce larger transcoding cost as there exists transcoding redundancy. Figure 18 shows the data traffic comsumed in the cloud and the edge. We find that CI-CSNN consumes less data traffic than C-nI in the cloud, as CI-CSNN focuses on not only increasing the bitrate but also improving the interaction experience. As PIECE can explore more on the edge devices, the data traffic utilization is also larger compared to the CE-nI method.

After the above analysis on the viewer's overall QoE and the costs of different methods, we further focus on the efficiency of the methods, i.e., how much viewer's QoE can be achieved with unit cost. We set the price of transcoding and delivery in the cloud as unit price and the price in the edge is p. The efficiency is defined as the achieved average QoE divided by the consumed cost. The results of transcoding and delivery efficiency are provided in Figure 19 and Figure 20, respectively. The efficiency of methods using the cloud-edge architecture decrease with the edge price p. Note that $p = 1$ is a special case that the price in the cloud equals that in the edge, in which case our method achieves the highest efficiency as for the transcoding cost as well as the delivery cost. In the future network, the price of the edge network is promising to decrease, which will further benefit the cloud-edge architecture.

8 CONCLUSION

This paper addresses the challenges of the video delivery problem in crowd-interactive livecast, resulting from the heterogeneous viewer interaction pattern. By conducting extensive measurement of traces obtained from the Twitch system, we observe unique characteristics related to viewers, broadcasters, and viewer-broadcaster pairs. Based on the observations, we design a deep neural network model to capture the viewer interaction pattern. Specifically, we employ a data preprocessing method considering continuous values and categorical information, and design a cost-sensitive neural network to cope with the data imbalance problem. Sequently, for the first time, we study the interaction QoE in the CIL service. In order to realize flexible network management, we design an efficient path selection algorithm in the cloud-edge architecture. Experiments on real traces further demonstrate the superiority of our design, which can improve the average viewer's QoE by 26%. Specifically, the startup delay is reduced by 30% and the interaction QoE is improved by 21%.

9 ACKNOWLEDGEMENT

The work of Haitian Pang and Lifeng Sun is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61472204, and 61521002, Beijing Key Laboratory of Networked Multimedia under Grant No. Z161100005016051, and Key Research and Development Project under Grant No. 2018YFB1003703. The work of Zhi Wang is supported by the NSFC under Grant No. 61531006 and SZSTI. The work of Jiangchuan Liu is supported by a Canada NSERC Discovery Grant and an NSERC E.W.R. Steacie Memorial Fellowship. The work of Haitian Pang is partly supported by Jiangxing Inc..

REFERENCES

[1] Qiyun He, Jiangchuan Liu, Chonggang Wang, and Bo Li. 2016. Coping with heterogeneous video contributors and viewers in crowdsourced live streaming: A cloud-based approach. *IEEE Transactions on Multimedia* 18, 5 (2016), 916–928.

[2] Qiyun He, Cong Zhang, and Jiangchuan Liu. 2017. CrowdTranscoding: Online Video Transcoding With Massive Viewers. *IEEE Transactions on Multimedia* 19, 6 (2017), 1365–1375.

[3] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computingâĂŤA key technology towards 5G. *ETSI white paper* 11, 11 (2015), 1–16.

[4] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, et al. 2016. Via: Improving internet telephony call quality using predictive relay selection. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 286–299.

[5] Baochun Li, Zhi Wang, Jiangchuan Liu, and Wenwu Zhu. 2013. Two decades of internet video streaming: A retrospective view. *ACM transactions on multimedia computing, communications, and applications (TOMM)* 9, 1s (2013), 33.

[6] Zhengye Liu, Yanming Shen, Keith W Ross, Shivendra S Panwar, and Yao Wang. 2008. Substream trading: Towards an open P2P live streaming system. In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*. IEEE, 94–103.

[7] Ming Ma, Lei Zhang, Jiangchuan Liu, Zhi Wang, Haitian Pang, Lifeng Sun, Weihua Li, Guangling Hou, and Kaiyan Chu. 2018. Characterizing User Behaviors in Mobile Personal Livecast: Towards an Edge Computing-assisted Paradigm. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 3s (2018), 66.

[8] Matthew K Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. 2015. Practical, real-time centralized control for cdn-based live video delivery. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 311–324.

[9] Haitian Pang, Zhi Wang, Chen Yan, Qinghua Ding, and Lifeng Sun. 2017. First Mile in Crowdsourced Live Streaming: A Content Harvest Network Approach. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*. ACM, 101–109.

[10] Haitian Pang, Zhi Wang, Chen Yan, Qinghua Ding, Kun Yi, Jiangchuan Liu, and Lifeng Sun. 2018. Content Harvest Network: Optimizing First Mile for Crowdsourced Live Streaming. *IEEE Transactions on Circuits and Systems for Video Technology* (2018).

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[12] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*. IEEE, 1–9.

[13] Lifeng Sun, Haitian Pang, and Lin Gao. 2018. Joint Sponsor Scheduling in Cellular and Edge Caching Networks for Mobile Video Delivery. *IEEE Transactions on Multimedia* (2018).

[14] vCPU. http://whatis.techtarget.com/definition/virtual-CPU-vCPU.

[15] Bo Yan, Shu Shi, Yong Liu, Weizhe Yuan, Haoqin He, Rittwik Jana, Yang Xu, and H Jonathan Chao. 2017. LiveJack: Integrating CDNs and Edge Clouds for Live Content Broadcasting. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 73–81.

[16] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. 2009. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 25–34.

[17] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 325–338.

[18] Cong Zhang, Jiangchuan Liu, and Haiyang Wang. 2016. Towards hybrid cloud-assisted crowdsourced live streaming: measurement and analysis. In *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 1.