When Cloud Meets Uncertain Crowd: An Auction Approach for Crowdsourced Livecast Transcoding

Yifei Zhu
School of Computing Science
Simon Fraser University, Canada
yza323@sfu.ca

Jiangchuan Liu
School of Computing Science
Simon Fraser University, Canada
jcliu@cs.sfu.ca

Zhi Wang
Graduate School at Shenzhen
Tsinghua University, China
wangzhi@sz.tsinghua.edu.cn

Cong Zhang
School of Computing Science
Simon Fraser University, Canada
congz@sfu.ca

ABSTRACT

In the emerging crowdsourced livecast services, numerous amateur broadcasters livestream their video contents to worldwide viewers and constantly interact with them through chat messages. Live video contents are transcoded into multiple quality versions to better service viewers with different network and device configurations. Cloud computing becomes a natural choice to handle these computational intensive tasks due to its elasticity and the "pay-as-you-go" billing model. However, given the significantly large number of concurrent channel numbers and the diverse viewer geo-distributions in this new crowdsourced livecast service, even the cloud becomes significantly expensive to cover the whole community and inadequate in fulfilling the latency requirement. In this paper, after observing the abundant computational resources residing in end viewers, we propose a Cloud-Crowd collaborative system, C2, which combines end viewers with cloud to perform video transcoding in a cost-efficient way. To quantify the heterogeneity and uncertainty of viewers and pass the asymmetric information barrier, we incorporate statistical descriptions into our bidding language and design truthful auctions to recruit stable viewers with appropriate incentives. We further tailor redundancy strategies for workloads with different Quality of Service requirements to improve the stability of our system. Desirable economic properties, like social efficiency, ex-post incentive compatibility, individual rationality, are proved to be guaranteed in our studied scenarios. Using traces captured from the popular Twitch platform, we show that C2 achieves up to 93% more cost saving than a pure cloud-based solution, and significantly outperforms other baseline approaches in both social welfare and system stability.

KEYWORDS

Crowdsourced livecast transcoding; auction mechanism; cloud computing; edge computing; uncertainty

MM'17, October 23–27, 2017, Mountain View, CA, USA.
 ISBN 978-1-4503-4906-2/17/10…$15.00
DOI: https://doi.org/10.1145/3123266.3123384

1 INTRODUCTION

The advances of personal computing devices and the prevalence of high-speed Internet access have pushed video streaming over the Internet into a new era. The crowdsourced livecast service (CLS) platforms, represented by Twitch, Periscope, and so on, have emerged in the market and have achieved tremendous success in recent years. In such services, numerous amateur broadcasters lively stream their video contents to viewers around the world. Fellow viewers watching the same channel constantly interact with each other and the channel broadcaster through chat messages. Twitch alone supported 9.7 million active users per day, hosted over two million broadcasters per month, and witnessed 14.2 billion chat messages in 2016 [6].

Similar to other traditional video streaming services, CLS providers transcode the same video content into different quality versions, and distribute appropriate versions to viewers to better match their varying network conditions and provide the best possible user experience [10]. Because these transcoding tasks are extremely CPU-intensive and require significant hardware, cloud becomes a natural choice for conducting such tasks due to its elasticity and the "pay-as-you-go" billing model. For example, major streaming hosting providers like Zencoder, Wowza, Ustream all provide their live streaming transcoding services based on public clouds.

The massive number of concurrent broadcasters, heterogeneous source contents and device configurations of end viewers in the CLS platforms generate substantially more codec and bit rate combinations for transcoding than classical content streaming services. As a result, this massive computational demand even makes a cloud-based solution prohibitively expensive. For instance, the existing Wowza Streaming Engine for transcoding charges as high as $2.77 per hour [3]. Consequently, covering 5% of Twitch peak concurrent broadcasters with one 150-minute session each would cost over $11,000 without considering other storage, bandwidth cost[1]. Therefore, in spite of Twitch's desire to offer transcoding services to the whole community, it instead offers transcoding services to a small portion of premium broadcasters and only extends to normal broadcasters when its capacity allows. In addition, the requirement on streaming latency in our studied CLS application is even more stringent due to the unprecedented level of interaction occurring in the

[1] In Twitch, top 5% broadcasters are regarded as influencers [2]; Peak concurrent broadcasters reach 35,610 [6]; Over 50% sessions last over 150 minutes [36].

community. High latency can severely affect the viewer-broadcaster, viewer-viewer interactive experience [32]. Yet considering the diverse geo-distribution of broadcasters, the closest cloud datacenter may still be far away from broadcasting sources, which inevitably increases network latency [33]. We thus seek a more cost-efficient, low-latency solution to cover more broadcasters.

Looking deeper into these CLS platforms, we observe the abundant computational resources residing in edge viewers. Advanced processors (e.g., Intel i7), powered with dedicated video transcoding core like Intel Quick Sync, are becoming the main stream setting for desktops, especially among high-end game devices. According to Steam, 50% of its PCs have 4 physical CPUs; 8.93% of processors operate at 3.7 GHz and above; 74.76% of PCs runs DirectX 12 GPUs [8]. The computing power of mobile devices is increasing even more surprisingly, like Apple's newly released A10 Fusion chip achieves close to workstation-level performance in the single-thread situation [5]; Google leverages mobile devices to run text recommendation training in its Gboard app [24]. As a successful game-oriented CLS provider, 65% of Twitch viewers come from desktop computers [30]. Even if we only focus on its high end devices, the transcoding performance of these devices (e.g., i7 4 GHz, 4 cores, 16 GB, built-in GPU with Intel QuickSync) is already comparable to, sometimes even better than, compute-optimized instances (e.g., `c4.8xlarge`) in the public cloud [4]. In addition, unlike traditional live streaming services where viewers of popular streams tend to be evenly distributed [21], geo-distribution of viewers for a specific broadcaster is highly skewed (48% of broadcasters have their viewers totally in the same province/state [23]) in our studied CLS platform. Since most viewers that consume a channel are located in the same region as the broadcaster, allowing local viewers to transcode streams will greatly reduce the communication distance from broadcasters to viewers. Inspired by this observation and the massive viewer base in these services, combining these viewers (crowd) with the cloud to do transcoding seems to be a promising solution.

However, unique challenges emerge if we want to conduct transcoding on a cloud-crowd system. Since transcoding inevitably costs extra power and bandwidth consumption, viewers' willingness to take on such tasks varies from person to person. Though we notice the formation of channel-oriented online communities, involving these viewers into taking computational intensive transcoding tasks still needs much stronger incentives to guarantee eventual performance. Thus we plan to recruit qualified viewers to take on these tasks and incentivize them with monetary rewards. Providing this incentive in a cost-efficient way requires the valuations of viewers for taking these tasks, which unfortunately are private and depend on their own profiles in most cases. In addition, even if they accept these tasks, their uncertainty in completing the tasks presents another challenge since viewers may not guarantee continuous transcoding for the whole live session or fail to complete the task due to their heterogenous computing power. What is worse, quantifying these uncertain behaviours may also require privacy information which creates cost as well; namely, this uncertainty information is also private. The heterogeneity and uncertainty of viewers as well as their private status all present serious challenges in viewer selection and pricing.

Therefore, in this paper, motivated by the abundant computational resources residing in edge viewers and the possible close distance from viewers to broadcasters in CLS platforms, we propose a cloud-crowd collaborative system, C2, to provide transcoding services at lower costs and with smaller delays. To pass the asymmetric information barrier in cost and uncertainty, we incorporate their statistical descriptions into our bidding language, design truthful auctions to recruit stable viewers and reward them with the right amount of incentive. To further offer reliable services, we devise redundancy strategies accordingly to different workload types. The mapping we present from our studied social welfare maximization problem to the min-cost flow problem not only guarantees optimal social efficiency but also is capable of absorbing different capacity requirements. We prove theoretically that our designed mechanisms guarantee social efficiency, incentive compatibility, and individual rationality in expectation. To the best of our knowledge, this is the first work to incorporate uncertainty directly into such hybrid transcoding architecture and study how applying redundancy might help. Large scale simulations driven by real traces demonstrate that our system can provide reliable transcoding services at low costs and achieve social efficiency.

The remainder of this paper proceeds as follows: Section 2 discusses the challenges for having a cloud-crowd collaborative system. Section 3 presents the formal formulation of our problem and other desirable goals. Section 4 presents the detailed mechanism design for broadcasting workloads. Section 5 extends our mechanism to handle pre-recorded video workloads. We evaluate our design through trace-driven simulations in Section 6, followed by related works in Section 7. Section 8 concludes the paper.

2 CHALLENGES AND PRINCIPLES

Most CLS platforms start primarily as a place to watch livestreamed contents and recently have began to diversify their content sources. Take Twitch for example, broadcasters normally livestream their game content to viewers and interact with them through chat messages. Besides these *broadcasting workloads*, Twitch also enables "Uploads" in late 2016, allowing broadcasters to upload pre-recorded content and publish later [7], to which we refer as *pre-recorded video workloads*. For both of types of workloads, the heterogeneous source formats and network/viewer device configurations require videos to be transcoded into multiple representations. For broadcasting workloads, each live session requires non-stop transcoding service with low latency, so that viewers can receive their desired video formats with small startup delay and keep interacting with broadcasters. Pre-recorded workloads, however, are interruption-tolerant as long as the user-defined playback deadline is met. The diversity of service types with a doubt increases the demand for computational resources and calls for more efficient cost management to meet heterogeneous Quality of Service (QoS) requirements.

Although assigning transcoding tasks to viewers may help reduce cost and latency, involving these viewers actually presents unique challenges for each workload type. First, to lower the acceptance barrier of such system, we envision that viewers would only do transcoding when they are surfing CLS platforms, which means that their duration for transcoding varies. Unlike entirely relying on cloud where instances are guaranteed by strict Service Level

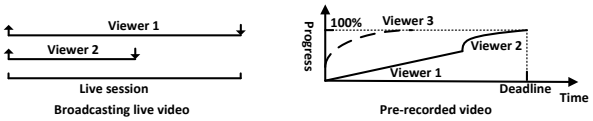


Figure 1: Redundancy illustration: simultaneous redundancy for broadcasting live video workloads and sequential redundancy for pre-recorded video workloads

Agreements (SLA), analysis of our trace shows that the duration of viewers for staying in the platform follows Pareto distribution with varying parameters. This heterogeneous participation behaviour greatly challenges the broadcasting transcoding workloads, since assigning tasks to unstable viewers definitely makes the transcoding inconsistent. Second, task execution performances of viewers (e.g., total task execution time) greatly vary due to their heterogeneous computing power. The diverse task execution time greatly challenges pre-recorded video workloads since assigning tasks to random viewers may make tasks miss their deadlines.

The redundancy principle has been widely applied to mitigate uncertainty and improve system performances in distributed systems [17, 28, 29, 31]. In our scenario, we devise different redundancy strategies for different types of workloads. Essentially, we recruit redundant viewers simultaneously to mitigate viewers' uncertainty in broadcasting workloads and recruit redundant viewers sequentially to mitigate viewers' uncertainty in pre-recorded video workloads. For instance, as illustrated in Fig.1, for a broadcasting task in the left subfigure, assume that each viewer i has its own probability p_i for leaving the system within a given session period. Introducing one more redundant viewer, viewer 2, to work along with the initial viewer, viewer 1, changes the probability of successfully transcoding a live session without interruption from $1 - p_1$ to $1 - p_1 p_2$. For pre-recorded video workloads in the right subfigure, assume that each worker has an individual probability for finishing the task before deadlines reflecting its heterogeneous computing power and load situations. Assigning a pre-recorded video task to a viewer, viewer 1, with smaller capability (usually with smaller cost as well) first, followed by a more powerful one, viewer 2, may be more cost efficient than simply assigning this task to the viewer with the strongest computing power, viewer 3.

3 SYSTEM MODEL AND PROBLEM FORMULATION

3.1 System Model

Globally, our system is divided into multiple regions, where each region has its own regional datacenter (referred to as "ingress server") for ingesting source videos, assigning transcoding tasks to viewers or cloud. The streaming segments after being transcoded are forwarded to CDN for further delivery. Fig. 2 illustrates the overall design of our cloud-crowd transcoding system. Specifically, source broadcasting contents are first collected by the ingress server through protocols such as RTMP (Real Time Messaging Protocol). Several viewers will then be selected for transcoding according to

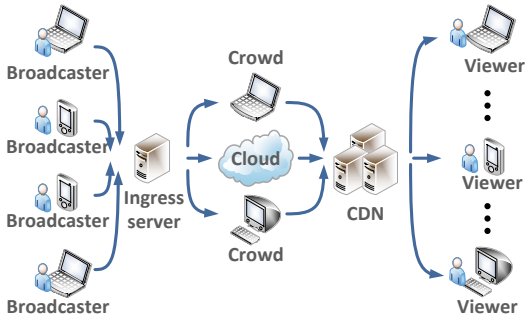


Figure 2: Cloud-Crowd system overview

certain criteria. Unmatched tasks after this selection or during the broadcasting process will be sent to cloud if no further satisfiable transcoding viewers can be found locally. The selected viewers and cloud servers transcode video contents into different quality versions and send them to CDN serving all viewers.

A live session with source format 1080p60fps will be transcoded into 720p60fps, 720p30fps, 540p30fps, etc., according to current twitch settings. Each representation of this live session represents a task that needs to be transcoded in our scenario. We leverage similar streaming test module like Twitch Inspector[2] in the existing Twitch platform to evaluate the network performance of viewers. Only viewers passing the minimum bandwidth requirement can be added into the candidate pool. After this initial capability evaluation, we have a viewer set $N = \{1, 2..., |N|\}$ and a transcoding task set $M = \{1, 2, ..., |M|\}$ in our system. Let $c^l_{i,j}$ denote the cost of viewer $i \in N$ for taking a transcoding task $j \in M$ in broadcasting workloads, and $p^l_{i,j}$ denote the probability for failing to complete task j, namely viewer i leaves the platform before the end of its serviced transcoding task j. Similarly, we have $c^d_{i,j}$ for the cost of viewer i in taking the pre-recorded video task j and $p^d_{i,j}$ for the probability of completing its assigned task before the deadline. The cost of viewers reflecting their willingness to accept the task can be influenced by various factors, like their regional electricity price, making their willingness vary from person to person.

Each viewer thus has a type $\theta_i = (c^\tau_i, p^\tau_i)$ where $\tau \in \{l, d\}$ represents the total type set of workload. Specifically, $(c^l_i, p^l_i) = \{(c^l_{i,j}, p^l_{i,j}), \forall j \in M\}$ for broadcasting tasks; $(c^d_i, p^d_i) = \{(c^d_{i,j}, p^d_{i,j}), \forall j \in M\}$for pre-recorded video tasks. Let $\hat{\theta}_i = (\hat{c}^\tau_i, \hat{p}^\tau_i)$ denote the declared type (bid) of viewer i since selfish viewers may misreport their types to gain better utility. Let $\theta_{-i} = (\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_N)$ be the profile types of all viewers except i, and (θ_i, θ_{-i}) completes the whole profile of all viewers, θ. We adopt the direct revelation principle in mechanism design to design auctions since it provides clear input information and allows us to focus on devising direct mechanisms in our auction. Following the direct revelation principle, a mapping function π, which includes an allocation mechanism (also known as social choice function) f and a payment mechanism λ, maps collected types to result R. $\theta_1 \times \theta_2 \times ... \times \theta_n \rightarrow R$ in our auction mechanism. Given one collected type input, the allocation and

[2]https://inspector.twitch.tv

Table 1: Table of important notations

Symbol	Definition
π	auction mechanisms, including f and λ
f, λ	allocation policy, payment policy
τ	workload type set
$\theta_i = (p, c)$	type of viewer i, including uncertainty p and cost c
V_j	valuation for task j
$\overline{w}(\pi(\theta))$	expected social welfare under current profile type θ
B_j	redundancy capacity of task j
$\boldsymbol{\gamma}$	eventual execution result
$c_{i,j}$	cost of viewer i in taking task j

payment results of all viewers, $(f_1, f_2, ...f_{|N|}, \lambda_1, \lambda_2, ..., \lambda_{|N|})$, form one resulting result $r \in R$. Since the selected viewer may leave the transcoding task during the transcoding process or fail to complete the transcoding task before the deadline probabilistically, we denote the eventual task execution result as a vector $\boldsymbol{\gamma}$, where $\gamma_j \in \boldsymbol{\gamma}$ is 1 if the task j allocated to viewers is completed, 0 otherwise. The utility for viewer i follows commonly used quasi-linear utility form and is denoted as $u_i = \mathbf{E}\{\lambda(\theta_i, \boldsymbol{\gamma}) - c_i(f_i(\theta_i))\}$. Utility for the ingest server, acting as the auctioneer, is $\mathbf{E}\{\sum_{j\in M} V_j(\theta_i, \boldsymbol{\gamma}) - \sum_{i\in N} \lambda(\theta_i, \boldsymbol{\gamma})\}$. The valuation for a transcoding task j, V_j, can be defined as any valuable metrics that are important to service provider measured in currency, like possible revenue of this channel. The social welfare in turn is $\overline{w}(\pi(\theta)) = \mathbf{E}\{\sum_{j\in M} V_j(\theta, \boldsymbol{\gamma}) - \sum_{i\in N} c_i(f_i(\theta_i))\}$. Social welfare can be regarded as a generalization of common performance metrics, such as utilization, to a setting with utility-weighted tasks and operation costs.

A good mechanism is expected to satisfy social efficiency, individual rationality and incentive compatibility. We summarizes important notations in Table 1 and present the formal definition of these desirable properties in the following for clarification and proof purpose.

Definition 3.1. A mechanism is ex-post incentive compatible, if for every bidder i and valuation type $\theta_i = (c_i^\tau, c_i^\tau)$, declaring their true type θ_i is the best response given all other players declare their true type θ_{-i}. Namely,

$$u_i(\pi(c_i^\tau, c_{-i}^\tau, p_i^\tau, p_{-i}^\tau)) \geq u_i(\pi(\hat{c}_i^\tau, c_{-i}^\tau, \hat{p}_i^\tau, p_{-i}^\tau))$$

Definition 3.2. A mechanism is ex-post individual rationality, if for every bidder i, any profile type $\theta_i = (c_i^\tau, c_i^\tau)$ and θ_{-i}, its expected utility is always non-negative if being selected. Namely,

$$\mathbf{E}\{u_i(\pi(c_i^\tau, c_{-i}^\tau, p_i^\tau, p_{-i}^\tau))\} \geq 0$$

Definition 3.3. A mechanism achieves ex-post social efficiency, if, given any profile types θ , its allocation policy f^* maximizes the sum of utilities of all bidders and auctioneer in the system in expectation. Namely,

$$\mathbf{E}_{f^*}\{\overline{w}(\pi(\theta_i, \theta_{-i}))\} \geq \mathbf{E}_{f'}\{\overline{w}(\pi(\theta_i, \theta_{-i}))\}$$

3.2 Social Welfare Maximization for Cloud-Crowd Collaboration

In our studied cloud-crowd collaborative system, we aim at maximizing the system wide utility, social welfare, in expectation given the private cost and uncertainty for finishing different type of workloads. To help describe our formulation, we introduce a set of 0-1 variable $x_{i,j}^\tau$ for each pair of viewer and task. Variable $x_{i,j}^\tau$ equals one if viewer v_i is assigned for task j of type τ. B is maximum number of viewers allowed for each task j. We formally formulate our problem in the following:

$$\max \quad \mathbf{E}\{\sum_{j\in M} V_j(\theta, \boldsymbol{\gamma}) - \sum_{i\in N} c_i(f_i(\theta_i))\} \tag{1}$$

$$\text{s.t.} \quad \sum_i x_{i,j}^\tau \leq B_j, \forall j \in M, \forall \tau \tag{2}$$

$$x_i \in \{0, 1\}, \forall i, B_j \in \mathbb{Z}_+^*, \forall j \tag{3}$$

Constraint (2) denotes that the number of viewers selected for a task j should be smaller than the required redundancy capacity; Previous works in cloud-based transcoding propose methods to estimate the transcoding resources needed by each transcoding task, and assign tasks to instances within their capability during the task assignment process[10]. Thanks to the abundant candidate pool from the crowd, we design our system to require each viewer only transcode one task of one type at one time to simplify the complex estimation process. If we choose to multiplex multiple tasks on a single viewer, we can easily relax this constraint to other values which does not affect the mechanism we proposed to get the optimal solution.

4 MECHANISM DESIGN FOR CLOUD-CROWD COLLABORATIVE TRANSCODING

In this section, we demonstrate how to design mechanisms to handle broadcasting workloads and apply the redundancy principle to improve social welfare further. For broadcasting workloads, each live session requires non-stop transcoding service with low latency, so that viewers can constantly receive their desired video formats. To guarantee this continuity in transcoding service, it is suboptimal to just select one viewer for transcoding one representation of a channel and start selecting another new viewer after the previous viewer abandons task abruptly. In contrast, having multiple viewers transcode one representation of a channel simultaneously can greatly improve social welfare and guarantee availability of the live session at all representations.

Following direct revelation principle, if we want to involve uncertain viewers into our transcoding process, explicitly asking for their probabilities of completing those transcoding tasks seems to be the most straightforward approach. However, self-interested viewers may misreport their probability, claiming to be more competent than they really are, to win the rewards. Even worse, the classical auction mechanism, like VCG mechanism, cannot be directly applied to our scenario to guarantee truthfulness in bidding due to its deterministic nature. Naively adding probability into it can jeopardize the truthfulness of the whole mechanism [26]. In our game theoretic scenario, our objective hence is to design a practical auction mechanism to guarantee desirable economic properties, like incentive compatibility in cost and likelihood for finishing the transcoding and maximize social welfare at the same time.

Our mechanism, presented in Algorithm 1, has two parts: the allocation mechanism and the payment determination mechanism. For the allocation part, we select viewers to transcode our tasks so

that the expected social welfare of the whole system can be maximized. We need to solve our problem optimally, since any non-exact solution can harm the incentive compatibility and individual rationality of our mechanism as we will see in the proof. After observing that every scheduling decision binds with a quantifiable cost, we plan to adopt a graph-based declarative description to our problem. However, finding the appropriate graph structure to encode the constraints and our goals is not that straightforward. Bipartite matching with N and M on each side can describe one-to-one channel-to-task assignments, but fails to capture the redundancy design in our mechanism. By adding another set of vertex other than source/sink, N, and M, we manage to represent our social welfare maximization problem into a min-cost flow problem.

We refer to the viewers that being selected for a task as the viewer group setting for this task. Given a set of viewers N, a set of transcoding task M, and the constraints defined in formulation, we construct a flow network $G = \{N \cup D \cup M \cup \{s, t\}, E\}$, where D represents possible combinations for all tasks $(v_1, v_2, ..., v_{|B|}), \forall v_i \in N$. The supply of s and the demand of t are all $|M|$ in value. The capacity of any edge between D and M is set to be 1. The cost of any edge between D and M is set to be $-\overline{u}_e = V_j(1 - \prod p^l_{i,j}) - \sum_i c_{i,j}, \forall j \in N, \forall i \in D$. Notice that normally a min-cost flow problem works when all costs are non-negative. We can transform edges with negative cost to edges with non-negative cost using edge reversal transformation techniques. After transforming to min-cost flow problem, based on the integral flow theorem [9], we know that our min-cost flow problem can be solved optimally. Equivalently, our social welfare maximization problem can be optimized, given the truthful telling from viewers.

In practice, we can improve the availability of a channel further by leveraging stable cloud instances. For example, cloud instances can take on tasks that correspond to the lowest quality representation of a channel and the rest higher quality representations are left to the powerful viewers. By doing this, even failures in all selected transcoding viewers of the channel will not lead to total channel unavailability. The involvement of cloud enables us to guarantee this availability which is crucial for user experience in our application and cannot be easily solved by previous pure peer solutions. We can also trim the edges between D and M constructed in previous networks by only assigning a transcoding task to viewers who follow or show preference to this broadcaster.

Achieving incentive compatibility in the uncertainty of viewer behaviour thus becomes the key challenge of our whole mechanism. We solve this by binding payment for a viewer with the actual outcome of its assigned transcoding task, namely, whether the allocated task for this viewer has been successfully transcoded or not. Specifically, the payment for a selected viewer i would be the expected social welfare of others (excluding the cost of i) minus the expected social welfare without the existence of viewer at all, $\overline{w}(\pi(c^l_{-i}, p^l_i, \theta_{-i})) - \overline{w}(\pi(\theta_{-i}))$, if viewer i finished this task. The first item in this payment formula excludes the cost of its own to avoid manipulation. The dependence of payment on real execution result further guarantees incentive compatibility. If the selected task is not successfully transcoded, our mechanism incurs a penalty for this viewer, which equals the expected social welfare without the existence of viewer, $-\overline{w}(\pi(\theta_{-i}))$. This penalty is necessary for us to prove the incentive compatibility in uncertainty. Since some tasks may fail to find desirable viewers and uncertain viewers may still leave the task during the transcoding process, we use cloud to guarantee a continuous transcoding process. Cloud thus is treated as a special bidder in our system whose cost is a public information.

Algorithm 1 The auction mechanism for C2

```
1: //The allocation mechanism
2: for all task j in M do
3:    Construct a flow network G = {N ∪ M ∪ D ∪ {s, t}, E}
4:    Select B_j viewers for each task j based on the solution of min-cost
      flow problem arg min_{j∈N}(∑ cost(e)), ∀e ∈ E
5: end for
6: //The payment mechanism:
7: while a live session ends or a channel reaches its publishing deadline
   do
8:    if γ_j = 1 then
9:       λ_i = w̄(π(c^l_{-i}, p^l_i, θ_{-i})) − w̄(π(θ_{-i})), ∀i ∈ {i|f_i = j}
10:   else
11:      λ_i = −w̄(π(θ_{-i})), ∀i ∈ {i|f_i = j}
12:   end if
13: end while
```

Before the formal proof, we use a small, simplified example to help understand how our mechanism works in one task scenario, even though our previous mechanism is designed to work under multiple tasks scenarios. Suppose we have one broadcasting task of value 10, two viewers, A and B, with type (2, 0.3) and (4, 0.2) respectively. The redundancy level of task is 1. The optimal strategy thus is to choose viewer A with the expected social welfare 5 (social efficiency). The payment for viewer A is $10 - (0.8 \times 10 - 4) = 6$ if it succeeded, and -4 otherwise. Its expected utility becomes $0.7 \times 6 - 0.3 \times 4 - 2 = 0.1$, which is greater than 0 (individual rationality). Even if B lies about its probability as 0, it actually has a negative expected utility: $0.8 \times 5 - 0.2 \times 5 - 4 = -1$, which prevents it from doing that (incentive compatibility).

Theorem 4.1. *Our proposed auction mechanism guarantees social efficiency in expectation.*

Theorem 4.2. *Our proposed auction mechanism guarantees ex-post individual rationality.*

Theorem 4.3. *Our proposed auction mechanism guarantees ex-post incentive compatibility in expectation.*

Theorem 4.1 is obvious since we solve our problem optimally. We prove Theorem 4.2 by comparing the utility gained from truthful telling and misreporting. We prove Theorem 4.3 by constructing the targeted social welfare form by analyzing the expected utility of a viewer. See the tech report for their detailed proofs [1].

5 TRANSCODING PRE-RECORDED VIDEO IN C2

In this section, we demonstrate how our mechanism can handle pre-recorded video workloads with a different redundancy strategy. Unlike broadcasting videos, the uploaded short videos only need to be transcoded before a certain deadline. Specifically, we define the SLA on how quickly the uploaded videos need to be available in defined representations as the duration of a video/constant time [20].

The transcoding time of each video depends on various factors like its duration, complexity of scenes, computating power of viewers. Due to efficient performance isolation and mature virtualization techniques, most of previous works assume the performance of instances are homogeneous and stable. In contrast, viewers in C2 obviously have heterogenous computing power, not to mention that the competition with local workloads also makes the transcoding time even more unpredictable. This heterogeneous task execution time greatly challenges decision making in assigning pre-recorded video workloads. Most of previous works deterministically estimate the transcoding time to do further allocation, where methods range from simple linear model [19] to complex neural networks [14]. Considering the difficulty in deterministically estimating transcoding time in practice, Zhang et.al in [37] take empirical approaches that treat the transcoding time as a distribution. In this paper, we also assume the transcoding time on a viewer follows its own distribution from a statistical perspective, given its device configurations, a video task and its targeted resolution.

Recall that $c^d_{i,j}$ denotes the cost of viewer i in taking the pre-recorded video task j and $p^d_{i,j}$ for the probability of completing its assigned task before the deadline. Facing this private information, our goal is to select viewers to complete the task before the deadline. Since each selected viewer may finish the task any time and transcoding pre-recorded videos allows interruptibility, it is suboptimal to select redundant viewers to transcode the same video concurrently in the beginning anymore. On the contrary, since a viewer may not finish the task before the deadline, it could help if we invoke another viewer when we think that the previous viewer will miss the deadline. In short, our problem becomes which viewers to select, and when redundant viewers can be selected to help finishing the task. The order of viewers for each task also matters now, we thus denote the permutation of all viewer grouping setting options as D', replacing the combination result D in the previous broadcasting workloads. For pre-recorded video tasks, the expected social welfare is

$$\overline{w}(\pi(\theta_i, \theta_{-i})) = \sum_{j \in N}(V_j(1 - \prod(1 - p^d_{i,j})) - \sum_{i \in D'} c_{i,j} \prod_{k=1}^{i-1}(1 - p^d_{k,j}))$$

In our scenario, if the distribution of transcoding time is public information and has the memoryless property, like the exponential distribution, we may derive a closed form solution for invocation time given viewer group settings. However, it turns out that distributions of transcoding time do not necessarily have this property, like the Gamma distribution found in [37]. Without having distribution with memoryless property or even without knowing the distribution information of viewers at all, calculating which viewers to choose and when to select redundant viewers is NP-hard, while solving it sub-optimally endangers our incentive compatibility. To guarantee our mechanism is still incentive compatibility, we design our auction to operate in discrete time slots, where the duration of one time slot is determined according to the duration of channels. In addition, due to limited bandwidth capacity, management complexity, and diminishing marginal gain by having extra viewers, each task j in our system is upper bounded by B_j. These two settings in turn reduce the feasible solution space. According to theoretical results in [25], it is then possible to run the optimal algorithm on this restricted solution space to achieve the incentive compatibility. Based on this, we can now find out the optimal viewer group setting and the time to invoke redundant viewers in polynomial time. Calculating payment for each selected viewer still follows the policy we defined in the previous section only with different calculations for expected social welfare.

Theorem 5.1. *After reducing the solution space through time discretion and bounding redundancy level, our proposed auction mechanism still guarantees ex-post incentive compatibility in expectation.*

We prove this theorem by proving that our mechanism finds the maximal solution under restricted solution space according to [25]. See the tech report for the detailed proof [1].

Complexity analysis The known worst-case complexity bound on the min-cost flow problem for a graph with E edges and V nodes is $O\{ElogV(E + Vlog(V))\}$[27]. Since our system has far more viewers than broadcasting channels, the overall complexity for running the mechanism in broadcasting workloads is $O\{N^B log(N)\}$. Similarly, for pre-recorded video workloads, covering at most T time slots, we have the overall complexity equal $O\{|N|^B T^{B-1}\}$.

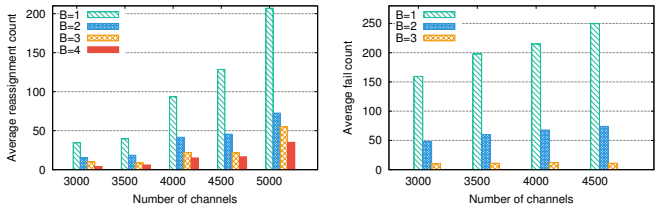
6 EVALUATION

6.1 Dataset and Experiment Settings

We collected the channel/broadcaster trace through Twitch's public API from January 25 to February 27, 2015. This public API provides the game name, viewer number, and some other related information of every broadcast channel. In the mean time, we further captured the channel-based viewer trace through connecting to the Internet Relay Chat (IRC) interface offered by Twitch. This trace contains over 10 million join/leave records of viewers in certain channels in the same period. We use the price of the default instance type, *m4.large*, as the cost for transcoding a task in the cloud. Bids are generated randomly between 0 and this price. The value of a channel is calculated proportional to its accumulated popularity, number of viewers watching this channel over the entire time span. Default SLA on transcoding deadline in pre-recorded video workloads is defined to be 1, namely a 150-minute duration video uploaded at time t is expected to be available in $t + 150$ time.

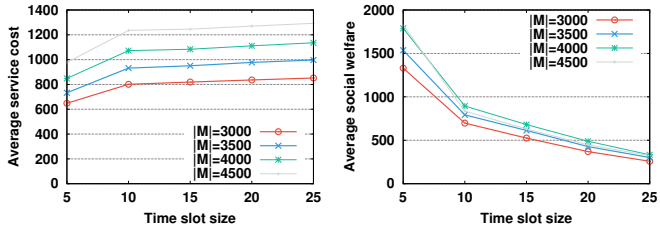
Uncertain behaviour For broadcasting workloads, the probability for transcoding an entire live session is derived from the Pareto distribution with its parameters varying from 0.5 to 0.9 as observed in our trace. This varying likelihood for completing the task reflects the heterogenous behaviour of viewers in taking our tasks. For pre-recorded video workloads, we follow the same statistical model as in [37] where transcoding time of a specified file is determined by its size and a random variable **t** reflecting the transcoding time of a unit size segments. The distribution of **t** can be modelled by a Gamma distribution function in their study, with the shape parameter ranges from 4.868 to 5.209, and the scale parameter ranges from 0.101 to 0.145. Since not all viewers are capable of taking the transcoding tasks, we sample only 1% of viewers from our enormous viewer pool to simulate the eligible viewers [3].

[3] According to the Steam and Twitch statistics mentioned in Sec. 1, roughly 2% Twitch users have PCs with 4-core CPUs operating at 3.7GHz and above, and DirectX 12 GPUs. Sampling 1% of viewers here is a conservative choice to test the performance of C2.



(a) Average reassignment count in broadcasting workloads (b) Average fail count in pre-recorded video workloads

Figure 3: Impact of redundancy level (varying B)



(a) Average service cost at different channels (b) Average social welfare at different channels

Figure 4: Impact of timeslot size

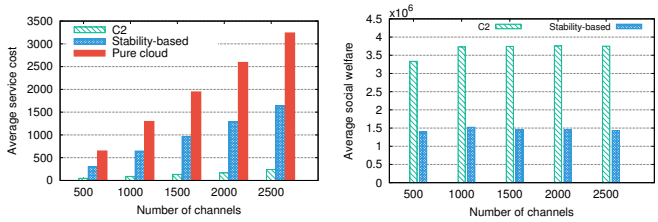
Baseline For broadcasting workloads, we compare our solution with a fully cloud-based approach and a stability-based method where viewers are selected greedily based on its average historical duration per cost [18]. Similarly, our baseline algorithm in pre-recorded video workloads is another greedy algorithm, greedy-efficiency, which selects viewers with the largest probability for finishing the task within the deadline per cost. We set a static price as thirty percent of the cloud counterpart to motivate viewers in the stability-based and greedy-efficiency approach.

Metrics To examine the performance of C2, we measure two important metrics: social welfare and total service cost for both two types of workloads. The first one is the ultimate optimization goal in our mechanism, reflecting the overall system efficiency. The second one is the overall cost for covering the transcoding demand of all channels. In addition, specifically for broadcasting workloads, we also measure reassignment count, which measures the number of times all transcoding viewers leave the assigned transcoding process. This reflects the stability of our system. Similarly, for pre-recorded video workloads, we measure the fail count, which denotes the number of times a channel missed the transcoding deadline.

6.2 Sensitivity Analysis

We first conduct sensitivity analysis to see the impact of time slot size and redundancy level on system performance, and then compare the performance of C2 with baseline approaches.

Impact of redundancy level Fig. 3 presents the impact of the redundancy level on different workload types. For broadcasting workloads in Fig. 3a , introducing one more redundant viewer can already reduce up to 65% reassignment than the baseline situation. This advantage sustains over 55% in the $B = 2$ cases. This gain diminishes as the redundant level increases. The average gain for increasing one more redundant viewers changes from 56% ($B = 1$ to $B = 2$) to around 25% ($B = 2$ to $B = 3$), and eventually drops to 5% ($B = 3$ to $B = 4$). For pre-recorded video workloads in Fig. 3b, introducing one more redundant viewer achieves around 70% less fail count reduction. The marginal benefit of introducing extra redundant viewers also decreases from 70% to 25% on average.



(a) Average service cost (b) Average social welfare

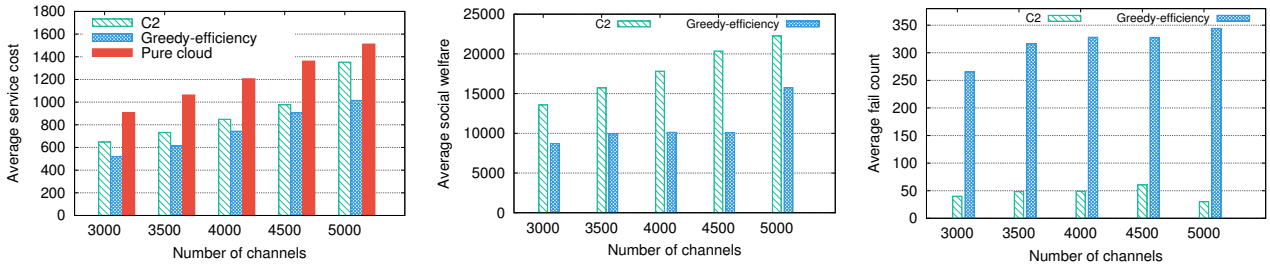
Figure 5: Performance comparison for broadcasting workloads (B = 2)

Impact of time slot size We next examine the impact of time slot size on service cost and social welfare in pre-recorded video workloads in Fig. 4. Service cost for transcoding the specified number of channels increases with the increase of time slot size. Correspondingly, social welfare of C2 decreases with the increase of time slot size. A smaller time slot means we can determine when to select redundant viewers in much finer granularity, which consequently leads to higher social welfare. Similarly, we also observe the decreasing of marginal performance gain in service cost and social welfare cases.

6.3 Comparison with Baseline Methods

Fig. 5a presents the service cost for transcoding specified numbers of channels in broadcasting workloads. C2's exploitation of normal viewers significantly reduces cost when compared with pure cloud implementation (e.g., by ≈ 93% compared to EC2 on-demand instance) and outperforms static pricing schemes used in stability approach (by ≈ 86%). This demonstrates the superiority of dynamic pricing in incentivizing viewers without introducing severe underpricing or overpricing problems. Fig. 5b demonstrates that C2 achieves 58% to 62% more social welfare than the stability-based approach. We omit the comparison of reassignment count in these situations due to the excess advantage in C2. The reassignment count of the stability-based approach ranges from 220 times in 480 channels to 877 times in 2500 channels. On the other hand, C2 just experiences reassignment 8 times in the 2500 cases. It is because that C2 does not rely on average duration history to filter reliable viewers, instead it directly elicits true probability of finishing the task from viewers.

For pre-recorded video workloads, C2 achieves over 28% service cost reduction in most cases than the pure cloud approach except when channel number reaches 5000 in Fig. 6a. The average service cost of C2 usually is around 10% more than that of greedy-efficiency



(a) Average service cost (b) Average social welfare (c) Average fail count(B=2)

Figure 6: Performance comparison for pre-recorded video workloads

approach. Notably, C2 in this setting procures two viewers for each transcoding task. With a little more investment in recruiting redundant viewers, C2 achieves at least 41% more social welfare than the greedy-efficiency approach in Fig. 6b. The social welfare gain even reaches 100% more than the greedy-efficiency approach when channel number is 4500. Thanks to the efficient invocation of redundant viewers, C2 is capable of finishing much more transcoding tasks than the greedy-efficiency approach in Fig. 6c.

7 RELATED WORKS

Attracted by the elasticity in computing power and the "pay-as-you-go" billing model, cloud naturally becomes the choice for supporting video transcoding services [10, 15, 22]. Aparicio-Pardo et al. in [10] study the appropriate target representations for transcoding to maximize viewers' satisfaction. Chen et al. in [11] propose a generic cloud renting framework to minimize leasing cost through service migration. Gao et al. in [15] present a dynamic resource provisioning algorithm to minimize the transcoding cost based on task preemption. Driven by the high cost in cloud transcoding, researchers have also studied novel architectures to help reduce transcoding cost [20, 34]. Krishnappa et al. in [20] outsource transcoding tasks to CDN and leverage online transcoding to improve user experience. The involvement of viewers brings unprecedented level of uncertainty and heterogeneity to computing resources which seldom appears in previous SLA guaranteed clouds. The altruistic assumption or naive fixed price incentive approaches also could not fully motivate users to take these computation-intensive tasks and truly reduce cost. In addition, peer transcoding in P2P systems shares some familiarity with our work in leveraging peer nodes to do transcoding [35]. However, these works mainly focus on how a tree structure can be constructed by peer information exchange in a distributed way. Our system still keeps a global control plane on task scheduling; essentially, each viewer still works as a powerful node to transcode alone. We focus on addressing the heterogeneity and uncertainty issues by using auction to do task allocation and incentive provisioning. What is more, cloud component is indispensable to our system due to its critical role in keeping stability and availability for our applications. Our solution echoes with the emerging paradigm of edge computing [16], while we do not rely on extra deployment of dedicated edge servers. Our mechanism also offers insights to scheduling in other distributed computing infrastructures with heterogenious computing power and under the control of different entities.

Auctions have been widely used to solve the resource allocation problem and incentive issues in varies application scenarios, like crowdsourcing [13], spectrum allocation [12], etc. It strives to elicit the private information from players through bidding and determines the appropriate reward as incentive. Since most previous auction mechanisms treat user behaviour as a static status, they are no longer applicable to our scenario due to the stochastic viewer computing behavior. We instead take a statistical description towards this uncertainty and incorporate it into our bidding language. There are few research works on addressing incentive problems in video streaming context. We tailor our mechanisms to explicitly meet the requirements of our studied applications. The most related work [18] studies a fully crowd transcoding approach. They select stable altruistic viewers based on the knowledge of viewer distribution. We propose a cloud-crowd solution to better improve system stability and assume no knowledge in viewers' distribution.

8 CONCLUSION

In this paper, we presented a novel transcoding system, C2, with a hybrid architecture which combines the computing power of cloud and crowd together to accomplish transcoding tasks in the emerging crowdsourced livecast systems. Facing the heterogeneity of viewers and the asymmetric information situation, we designed truthful auction mechanisms to select stable viewers for transcoding and tailor redundancy strategies for different types of workloads. We further proved theoretically that our proposed mechanism achieves social efficiency, individual rationality, and ex-post incentive compatibility. The trace-driven simulation demonstrated that our system achieves higher social welfare and lower service cost than the pure cloud solution.

ACKNOWLEDGMENTS

This research is supported by an Industrial Canada Technology Demonstration Program (TDP) grant, an NSERC Discovery Grant, and an E.W.R. Steacie Memorial Fellowship. Dr. Z. Wang's work is supported by the National Natural Science Foundation of China under Grant No. U1611461.

REFERENCES

[1] Technical Report. https://goo.gl/vDXrJ8.
[2] Twitch Audience Statistics. http://twitchadvertising.tv/audience/, 2015.
[3] Wowza Live Transcoding Engine. https://www.wowza.com/products/capabilities/live-transcoding, 2015.
[4] Wowza Transcoder Performance Benchmark. https://www.wowza.com/docs/wowza-transcoder-performance-benchmark, 2015.
[5] The iPhone's new chip should worry Intel. http://www.theverge.com/2016/9/16/12939310/iphone-7-a10-fusion-processor-apple-intel-future, 2016.
[6] Twitch Retrospective. https://www.twitch.tv/year/2016, 2016.
[7] Uploads Open Beta: Help shape the future of Uploads on Twitch. https://blog.twitch.tv/uploads-open-beta-now-live-aec0e1925989?sf37518783=1, 2016.
[8] Steam Hardware Survey: March 2017. http://store.steampowered.com/hwsurvey, 2017.
[9] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows: theory, algorithms, and applications. 1993.
[10] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon. Transcoding live adaptive video streams at a massive scale in the cloud. 2015.
[11] F. Chen, C. Zhang, F. Wang, and J. Liu. Crowdsourced live streaming over the cloud. In *Proc. IEEE INFOCOM*, 2015.
[12] Y. Chen, J. Zhang, K. Wu, and Q. Zhang. Tames: A truthful auction mechanism for heterogeneous spectrum allocation. In *Proc. IEEE INFOCOM*, pages 180–184, 2013.
[13] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos. Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *Proc. IEEE INFOCOM*, pages 1231–1239, 2014.
[14] G. Gao, H. Hu, Y. Wen, and C. Westphal. Resource provisioning and profit maximization for transcoding in clouds: A two-timescale approach. *IEEE Trans. Multimedia*, 2016.
[15] G. Gao, Y. Wen, and C. Westphal. Dynamic resource provisioning with qos guarantee for video transcoding in online video sharing service. In *Proc. ACM MM*, pages 868–877, 2016.
[16] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.
[17] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia. Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):347–360, 2015.
[18] Q. He, C. Zhang, and J. Liu. Crowdtranscoding: Online video transcoding with massive viewers. *IEEE Transactions on Multimedia*, 19(6):1365–1375, 2017.
[19] S. Ko, S. Park, and H. Han. Design analysis for real-time video transcoding on cloud systems. In *Proc. ACM SAC*, pages 1610–1615, 2013.
[20] D. K. Krishnappa, M. Zink, and R. K. Sitaraman. Optimizing the video transcoding workflow in content delivery networks. In *Proc. ACM MMSys*, pages 37–48, 2015.
[21] Z. Li, G. Xie, M. A. Kaafar, and K. Salamatian. User behavior characterization of a large-scale mobile live streaming system. In *Proc. ACM WWW*, pages 307–313, 2015.
[22] H. Ma, B. Seo, and R. Zimmermann. Dynamic scheduling on video transcoding for mpeg dash in the cloud environment. In *Proc. ACM MMSys*, 2014.
[23] M. Ma, L. Zhang, J. Liu, Z. Wang, W. Li, G. Hou, and L. Sun. Characterizing user behaviors in mobile personal livecast. In *ACM NOSSDAV*, pages 43–48, 2017.
[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.
[25] N. Nisan and A. Ronen. Computationally feasible vcg mechanisms. *J. Artif. Intell. Res.(JAIR)*, 29:19–47, 2007.
[26] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.
[27] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.
[28] N. B. Shah, K. Lee, and K. Ramchandran. When do redundant requests reduce latency? *IEEE Trans. Commu.*, 64(2):715–722, 2016.
[29] S. Stein, E. Gerdinga, A. Rogersa, K. Larson, and N. Jennings. Algorithms and mechanisms for procuring services with uncertain durations using redundancy. *Artificial Intelligence*, 175:2021–2060, 2011.
[30] Twitch. Twitch retrospective 2015. https://www.twitch.tv/year/2015, 2015.
[31] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Low latency via redundancy. In *Proc. ACM CoNEXT*, pages 283–294, 2013.
[32] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Y. Zhao. Anatomy of a personalized livestreaming system. In *Proc. ACM IMC*, pages 485–498, 2016.
[33] H. Wang, R. Shea, X. Ma, F. Wang, and J. Liu. On design and performance of cloud-based distributed interactive applications. In *Proc. IEEE ICNP*, pages 37–46, 2014.
[34] Z. Wang, L. Sun, C. Wu, W. Zhu, and S. Yang. Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming. In *Proc. IEEE INFOCOM*, 2014.
[35] J.-C. Wu, P. Huang, J. J. Yao, and H. H. Chen. A collaborative transcoding strategy for live broadcasting over peer-to-peer iptv networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(2):220–224, 2011.
[36] C. Zhang and J. Liu. On crowdsourced interactive live streaming: a twitch.tv-based measurement study. In *ACM NOSSDAV*, 2015.
[37] W. Zhang, Y. Wen, J. Cai, and D. O. Wu. Toward transcoding as a service in a multimedia cloud: Energy-efficient job-dispatching algorithm. *IEEE Trans. vehicular technology*, 63(5):2002–2012, 2014.