IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488868

A Universal Transcoding and Transmission Method for Livecast with Networked Multi-Agent Reinforcement Learning

Xingyan Chen* Changqiao Xu*† Mu Wang* Zhonghui Wu* Shujie Yang* Lujie Zhong‡ Gabriel-Miro Muntean§
* State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; ‡ Capital Normal University, Beijing 100048, China; § School of Electronic Engineering, Dublin City University, Dublin 9, Ireland, † corresponding author
Email: {chenxingyan, cqxu, wangmu, zhwu, ysj}@bupt.edu.cn, zhonglj@cnu.edu.cn, gabriel.muntean@dcu.ie

***Abstract*—Intensive video transcoding and data transmission are the most crucial tasks for large-scale Crowd-sourced Livecast Services (CLS). However, there exists no versatile model for joint optimization of computing resources (*e.g.*, CPU) and transmission resources (*e.g.*, bandwidth) in CLS systems, making maintaining the balance between saving resources and improving user viewing experience very challenging. In this paper, we first propose a novel universal model, called Augmented Graph Model (AGM), which converts the above joint optimization into a multi-hop routing problem. This model provides a new perspective for the analysis of resource allocation in CLS, as well as opens new avenues for problem-solving. Further, we design a decentralized Networked Multi-Agent Reinforcement Learning (MARL) approach and propose an actor-critic algorithm, allowing network nodes (agents) to distributively solve the multi-hop routing problem using AGM in a fully cooperative manner. By leveraging the computing resource of massive nodes efficiently, this approach has good scalability and can be employed in large-scale CLS. To the best of our knowledge, this work is the first attempt to apply networked MARL on CLS. Finally, we use the centralized (single-agent) RL algorithm as a benchmark to evaluate the numerical performance of our solution in a large-scale simulation. Additionally, experimental results based on a prototype system show that our solution is superior in saving resources and service performance to two alternative state-of-the-art solutions.**

I. Introduction

Due to their increased interactivity and real-time user experience, **C**rowd-sourced **L**ivecast **S**ervices (CLS) such as Twitch [1] and Douyu [2] are among the most popular online entertainment services. The latest statistics indicate that Twitch has almost 5.76 million broadcasters monthly and an average of 1.94 million concurrent viewers in 2020 [3]. Additionally, the average daily active viewer number of Douyu reached 24.77 million in July 2020 [4]. Moreover, CLS is extending its presence to all aspects of life, including education and health care, especially during the global COVID-19 pandemic. However, the increased use of CLS also brings new challenges to current CLS systems. A CLS provider needs to continuously deliver millions of content items to viewers, as well as perform online transcoding of massive amounts of video into multi-quality resolutions to fit various configurations of networks and devices. Hence, CLS deployment requires intensive computing and large bandwidth support, which is very costly. According to Douyu's annual report, the bandwidth cost in 2019 alone was $88.3 million which is 10% of its total expenditure [5].

To alleviate the CLS system overhead, researchers suggested using transcoding optimization schemes [6]–[9]. For example, a Lyapunov-based solution was proposed to minimize the operational cost of the CLS system in the cloud [6]. The authors of [7] designed an online algorithm that minimizes the cost of the Cloud-based CLS system while ensuring Quality of Experience (QoE). In addition, many other computing paradigms such as involving edge and crowd computing are considered as alternative solutions [10]–[17]. H. Pang *et al.* [11] proposed a novel concept, called transcoding delivery path, which involves video streaming transcoded into multiple versions during the delivery path. Further, a deep neural network based algorithm was proposed to optimize the transcoding delivery path for CLS flows to achieve efficient transcoding and delivery over a cloud-edge infrastructure. Z. Wang *et al.* [12] provided a joint online transcoding and delivery solution by formulating a joint resources allocation problem as a conventional 0/1 knapsack problem. A greedy-based heuristic algorithm was proposed to solve the resulting NP-hard problem. Y. Zhu *et al.* [14] proposed a novel cloud-crowd solution by leveraging massive crowd computing resources to assist the cloud, thereby achieving a 93% saving in terms of cloud overhead. The cloud-crowd solution was further extended with an auction-based incentive scheme [16] to motivate the crowd contributing resources for large-scale CLS deployment.

CLS systems have evolved from using a centralized cloud approach to employing a distributed computing paradigm with hybrid resources of cloud, edge, and crowd. By integrating cost-effective cloud, low latency edge, and cheap crowd, the distributed computing paradigm is identified as a promising solution for large-scale CLS. However, existing solutions still have some limitations such as unsatisfactory QoE performance [6]–[9], difficulty in terms of large-scale deployment [6]–[12], and insufficient delivery capacity [13]–[17]. We believe the main reason for this situation is that there exists no versatile analysis model which can jointly capture the stochastic features of transcoding and delivery in CLS. This complicates the modeling and often makes the results suboptimal. To solve this problem, we expand the ideas of **U**niversal **M**ax-**W**eight (UMW) [18] and **U**niversal **C**omputing **N**etwork **C**ontrol (UC-NC) [19] proposed by E. Modiano *et al.* and introduce a novel universal **A**ugmented **G**raph **M**odel (AGM). By adding virtual nodes and links in the network topology, we represent video transcoding as a process of video transmission over virtual links. However, video delivery is a process of video transfer between real network nodes. Therefore, there is a need for joint optimization of video transcoding and transmission and

978-1-6654-0325-3/21/$31.00 ©2021 IEEE

in AGM this process is converted into a problem of routing video data between different virtual and real nodes.

Although, AGM simplifies the analysis and modeling of the joint optimization, the deployment of large-scale CLS still requires a highly scalable approach. Current solutions such as UMW [18] and UCNC [19] can obtain the optimal network routing for each flow, but they all rely on a central coordinator which may show inflexibility in large-scale CLS. In UMW, the central node needs knowledge of network topology and virtual queue-length in every time-slot to solve the shortest route problem. Although the authors give a heuristic solution, it lacks performance analysis. To address this challenge, we employ the **M**ulti-**A**gent **R**einforcement **L**earning (MARL) [20] approach, which has been recognized as a promising solution for distributed network routing [21]–[27]. For example, P. Sun *et al.* in [23] proposed a **D**eep **R**einforcement **L**earning (DRL)-based network control framework, called SINET for routing optimization in Software-Defined Networking. However, the operation of the system relies heavily on the central controller and the problem of large-scale deployment is still not solved. A fully distributed packet routing framework with multi-agent deep reinforcement learning is proposed in [26]. In this framework, each node owns independent long short term memory based deep neural network which can learn the routing features from the memory of historical routing action and estimate the overall expected reward function (Q value). L. Yang *et al.* [27] designed a MARL-based distributed routing protocol called DMARL to improve the energy-saving and routing reliability for underwater optical wireless sensor networks. By using a two-tier reward structure, a global reward and a distributed reward, this approach can achieve efficient network energy balancing and support distributed deployment of algorithms. However, the above two solutions are based on Q-Learning, which is poorly suited to a networked multi-agent scenario. The main issue is that, in the training process, the routing strategy of each network node is changing, and the environment becomes unstable from the perspective of any individual agent (such as when a node makes the same routing decision twice, it may get totally different rewards, since other nodes adopt different decisions) [28]. This leads to instability in learning and unusable experience replay, and makes Q-Learning use infeasible.

This paper introduces the Augmented Graph Model which transforms the joint allocation optimization of computing resources and transmission resources into a generalized network routing problem. Further, we employ networked MARL as a solution to solve the routing problem in a fully distributed manner. Due to the drawbacks of Q-Learning in the multi-agent environment, we introduce a novel **M**ulti-**A**gent **A**ctor-**C**ritic (MAAC) solution based on the policy gradient theorem achieving a more robust system. By the adversarial learning between the actor and the critic, they optimize the action policy and action-value function, respectively. In MAAC, each agent makes decisions individually in the actor step and exchanges its data with its one-hop neighbors for global consensus in the critic step. Compared with the current CLS solutions [6]–[17], our work provides a new perspective for solving the joint optimization of transcoding and delivery in large-scale CLS.

The main contributions of this paper are as follows: We introduce a universal AGM which simplifies the problem of joint optimization of video transcoding and transmission over virtual network links. We use a networked multi-agent Markov decision process based on **Dec**entralized **MDP** (Dec-MDP) to reformulate the above problem and propose networked MARL based solution to distributively solve it for large-scale CLS. To the best of our knowledge, this is the first attempt to employ networked MARL in CLS. We design MAAC algorithm, which enables the global dissemination of local information at each agent using one-hop information sharing. This enables all agents in the CLS system to reach a consensus estimate and achieve the maximum reward for the entire system.We evaluate our solution through a large-scale numerical simulation and prototype system experiments. The experimental results show that our algorithm outperforms the current solutions in terms of both saving resources and service performance.

II. System Model and Problem Formulation

In this section, we first introduce the system model. We illustrate AGM for a large-scale CLS system with cloud-edge-crowd integration. Finally, we formulate the joint resource allocation problem of transcoding and delivery based on AGM.

A. Network Model

We consider the network topology of large-scale CLS as an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ links. We define the set of broadcasters, **C**loud **S**ervers (CSs) or **E**dge **S**ervers (ESs) and the viewers as $\mathcal{V}_p,\mathcal{V}_t,\mathcal{V}_c \in \mathcal{V}$, respectively. We assume that the time is slotted, such as $\mathcal{T}=\{1,2,...\}$ and each node $u \in \mathcal{V}$ has transcoding capability. Since the node may process tasks of other applications, we define the available computing resource at time t as a time-varying variable $c_u(t)$. Thus, we have $c_u(t) \in [0, c_u^{max}]$ where c_u^{max} is the maximum computing resources of node u. The network link from node u to node v is defined as $(u,v) \in \mathcal{E}$, $u,v \in \mathcal{V}$. When a video flow passes through link (u,v), it consumes $w_{(u,v)}$ bandwidth resources per unit. Since video streaming is also subject to various cross-traffic from other applications, we define the available bandwidth of the link (u,v) as a random variable $c_{(u,v)}(t) \in [0, c_{(u,v)}^{max}]$ where $c_{(u,v)}^{max}$ is the maximum bandwidth.

B. Transcoding Model

We define the CLS library as $\mathcal{F}=\{f_1,f_2,...,f_N\}$ which consists of $|\mathcal{F}|$ different channels. For each channel, we assume that the set of video resolution is $\mathcal{B}=\{b_0,b_1,...,b_m\}$. We consider the original video as b_0 and the lowest quality one as b_m, so we have $b_1 > ... > b_m$. Since the higher resolution video can be transcoded to lower quality versions [29], [30], the video transcoding model can be described as a state transition of different versions. Thus, the order set for the n-th channel is denoted by $\mathcal{O}_n=\{b_0,b_1,...,b_m\}$ which can be represented as a directed graph $\mathcal{G}'(\mathcal{V}',\mathcal{E}')$, as shown in Fig. 1 (a). We can see that each vertex $v'_h \in \mathcal{V}'$ corresponds to a video quality b_h and each link, such as $(h-1,h) \in \mathcal{E}'$ represents the transcoding process from resolution b_{h-1} to b_h.

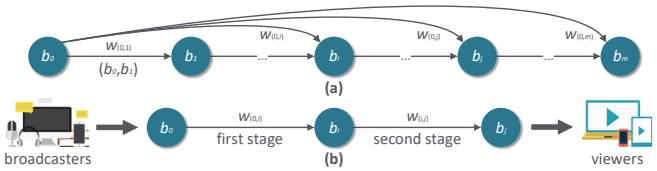

Fig. 1. (a) Illustration of the video transcoding in graph $\mathcal{G}'(\mathcal{V}', \mathcal{E}')$; (b) multi-stage transcoding process for a specific flow f.

When a video flow passes through link $(x, h), 0 < x < h$, we assume that the CLS system will consume $w_{(x,h)}$ computation resources per unit for transcoding. Fig. 1 (b) illustrates an example of a multi-stage transcoding process. In the first stage, the video flow f is transcoded from b_0 to b_i with computation requirement $w_{(0,i)}$ per unit. Further, the flow is converted into b_j in the second stage which consumes $w_{(i,j)}$ computing resources per unit. Thus, the total resource consumption per unit for the flow f is $w_f = w_{(0,i)} + w_{(i,j)}$.

C. Universal Augmented Graph Model

As shown in Fig. 2, we consider a large-scale CLS system with cloud-edge-crowd integration. The network topology of the system can be embodied as an overlay network. We define the overlay network as the original layer network enhanced with an undirected graph $\mathcal{G}_0(\mathcal{V}_0, \mathcal{E}_0)$ where $\mathcal{V}_0$ and $\mathcal{E}_0$ represent the nodes and links in original layer network, respectively. When video transcoding is invoked at a specific node, the resolution of the video will change, such as from b_0 to b_1. We define that for videos with different standard resolutions. Each resolution such as b_i corresponds to an augmented network layer $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i)$. Thus we have a layer graph $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i), i \in \{0, 1, ..., m\}$. Since the high resolution video can be transcoded into a lower one, we can connect the different layers with directed virtual links, such as the link from node 9_0 to node 9_1 in Fig. 2. We define the multi-layer graph $\mathcal{G}_{\boldsymbol{B}}(\mathcal{V}_{\boldsymbol{B}}, \mathcal{E}_{\boldsymbol{B}}), \mathcal{B} = \{0, 1, ..., m\}$ as the **A**ugmented **G**raph **M**odel (AGM). Besides, each link $l \in \mathcal{E}_{\boldsymbol{B}}$, is associated to a weight factor w_l which represents the resource (including computing resources and transmission resources) cost per unit.

We define three roles for nodes in the CLS system: *Consumer*, the viewer who issues a request for a specific channel; *Transcoder*, the intermediate node (CSs, ESs or viewers) which receives the video from the broadcaster or other *Transcoders* and converts them into the resolution asked by the *Consumer*; *Provider*, the broadcaster which continuously generates and uploads the original video to the CLS system. We can represent any patterns of transcoding and delivery for a flow with a route from *Provider* to *Consumer* on the AGM. For example, we consider that *Consumer* i and *Consumer* j request data from *Provider* at the same time. The two processes of the video access can be represented as two virtual paths (VPath i and VPath j) shown by the blue line and red line in Fig. 2. The blue line represents a video transcoding delivery path $(0_0, 2_0, 3_0, 5_0, 5_1, 7_1)$ from *Provider* to *Consumer* i. The transcoding process invoked at node 5 converts the video from b_0 to b_1. Meanwhile, for the red line, transcoding processes are invoked at node 5 and node 8. Notice that the red line and the blue line have overlapping paths. In this case, the CLS system only needs to transmit and transcode the video stream once on the overlapping path, reducing the system transmission overhead. In other words, when two VPaths overlap more, in general, the resource utilization is more efficient.

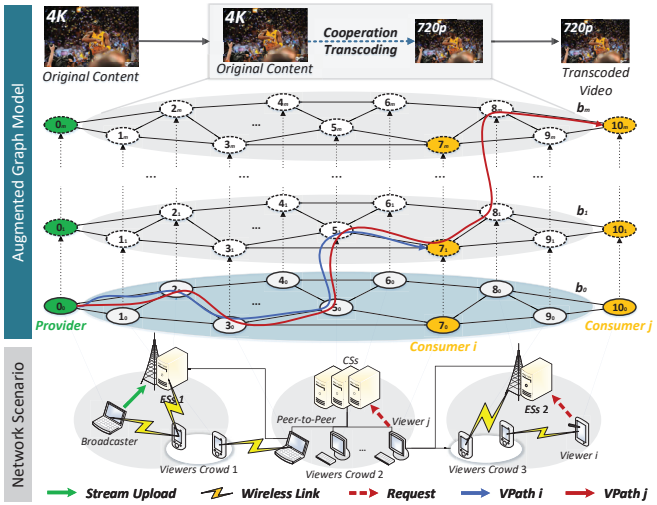

Fig. 2. Illustration of AGM for CLS based on Cloud-Edge-Crowd integration.

It can be seen how AGM transformed the joint resource allocation problem of transcoding and transmission into a network routing problem.

D. Problem Formulation

Our goal is to optimize Quality of Service and achieve the lowest joint resources overhead of the CLS system. First, we consider the service performance of CLS and define the received data unit of *Consumer* $c \in \mathcal{V}_c$ at time t as $x_c(t)$. When the received video resolution is b_c, the received data rate for *Consumer* c is equal to $b_c x_c(t)$. Thus, the utility of *Consumer* c is $U\left(b_c x_c\left(t\right)\right)$ where $U(\cdot)$ is the utility function. The utility function $U(\cdot)$ can have multiple forms, such as that in [31]. The overall QoS of CLS system is $\sum_{c \in \mathcal{V}_c} U\left(b_c x_c\left(t\right)\right)$.

We consider the joint resource overhead of the CLS system based on AGM. As mentioned earlier, when diverse video flows passes through link $l \in \mathcal{E}_{\mathcal{B}}$, it costs w_l resources per unit. Note that since node may receive multiple requests for same video version (the overlapping path), the node only needs to process once. Therefore, the resource consumption of link l is denoted as $\sum_{c \in \mathcal{V}_{c,l}} w_l x_c(t)$ where $\mathcal{V}_{c,l}$ represents the set of all *Consumer* with the flows of diverse videos passing through link l. The overall resources cost of the system can be written as $f_c(\boldsymbol{x}) = \sum_{l \in \mathcal{E}_{\mathcal{B}}} \sum_{c \in \mathcal{V}_{c,l}} w_l x_c(t)$, where $f_c(\cdot)$ is the cost function. By combining the service performance and joint resources cost, we formulate the joint resources allocation of transcoding and delivery for CLS as follows.

$$\forall t : \mathcal{U}_t(\boldsymbol{x}) = \max \left(\sum_{c \in \mathcal{V}_c} U(b_c x_c(t)) - \beta f_c(\boldsymbol{x}) \right) \tag{1a}$$

$$\text{s.t.} \sum_{l \in \mathcal{E}_{\mathcal{B}}(v)} \sum_{u \in \mathcal{V}_{c,l}} w_l x_c(t) \leq c_v(t), \ \forall v \in \mathcal{V} \tag{1b}$$

$$\sum_{l \in \mathcal{E}_{\mathcal{B}}(u,v)} \sum_{u \in \mathcal{V}_{c,l}} w_l x_c(t) \leq c_{(u,v)}(t), \ \forall (u,v) \in \mathcal{E} \tag{1c}$$

$$x_c(t) \in [0, x_{max}], \ \forall c \in \mathcal{V}_c \tag{1d}$$

where $\mathcal{U}_t(\boldsymbol{x})$ is the objective function with $\boldsymbol{x} = (x_1, ..., x_{|\mathcal{V}_c|})$, β is the weight factor, and x_{max} is the maximum reception rate. $\mathcal{E}_{\mathcal{B}}(v)$ represents the set of all virtual links for transcoding processes associated with node v and $\mathcal{E}_{\mathcal{B}}(u, v)$ is the set of all virtual transmission links corresponding to the real link (u, v). The objective (1a) is to maximize the sum of overall utility and the negative total consumption of virtual links. The constraint (1b) indicates that the transcoding rate cannot exceed the available computing capacity of node u. The constraint (1c) shows that the transmission rate should be less than the available bandwidth of the link (u, v). Finally, (1d) is the boundary constraints of data rate for flow $c \in \mathcal{V}_c$.

III. Networked Multi-Agent Reinforcement Learning For Joint Transcoding and Transmission Optimization

In this section, we first provide the necessary background about decentralized Markov Decision Process (Dec-MDP) and Actor-critic (AC) algorithm. We give a brief introduction of networked Multi-Agent MDP and reformulate problem (1) in the networked MARL framework.

A. Background

1) Decentralized Markov decision process: A Dec-MDP is defined as a 5-tuple $\mathcal{M} = (\mathcal{V}_{\mathcal{B}}, S, A, P, R)$ where

- $\mathcal{V}_{\mathcal{B}}$: a set of agents corresponding to the virtual nodes.
- S: a finite joint state space. In the CLS system, the joint state $\boldsymbol{s}$ contains the available resources $\{c_l(t)\}, \forall l \in \mathcal{V}_{\mathcal{B}}$.
- A: is the set of joint actions $\boldsymbol{a} =< a_1, ..., a_{|\mathcal{V}_{\mathcal{B}}|} >$. For agent u, the set of available actions is A_u. For our problem, a joint action is to find a route for a specific flow $x(t)$ from a *Provider* to a *Consumer* in AGM.
- $P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) : S \times A \times S \to [0, 1]$ is a transition probability function from state $\boldsymbol{s}' \in S$ to $\boldsymbol{s} \in S$ when action $\boldsymbol{a} \in A$ is taken.
- $R : S \times A \to R$ is the immediate reward function which is denoted as $R(\boldsymbol{s}, \boldsymbol{a}) = \sum_{u \in \mathcal{V}_{\mathcal{B}}} \mathbb{E}[r^u_{t+1} | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}]$, where r^u_{t+1} is the reward of network agent u at time $t+1$.

For all agents $u \in \mathcal{V}_{\mathcal{B}}$, the joint policy is a mapping $\boldsymbol{\pi} =< \pi_1, ..., \pi_{|\mathcal{V}_{\mathcal{B}}|} >: S \times A \to [0, 1]$ to maximize the expectation of cumulative reward. We define the expected long-term averaged reward return $\mathcal{R}(\pi)$ as the following equation.

$$\mathcal{R}(\pi) = \frac{1}{T} \lim_{T \to \infty} \sum_{t=1}^{T} R(\boldsymbol{s}, \boldsymbol{a}) \tag{2a}$$

$$\overset{(a)}{=} \frac{1}{T|\mathcal{V}_{\mathcal{B}}|} \lim_{T \to \infty} \sum_{t=1}^{T} \sum_{c \in \mathcal{V}_{\mathcal{B}}} \mathbb{E}(r^u_{t+1}) \tag{2b}$$

$$\overset{(b)}{=} \sum_{\boldsymbol{s} \in S, \boldsymbol{a} \in A} d_\pi(s) \pi(\boldsymbol{s}, \boldsymbol{a}) \bar{R}(\boldsymbol{s}, \boldsymbol{a}) \tag{2c}$$

where $d_\pi(s) = \lim_t \mathbb{P}(s_t = s|\pi)$ is the stationary distribution under policy π and $\bar{\mathcal{R}}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[\bar{r}^u_{t+1} | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}]$ is the averaged reward function of all agents where $\bar{r}^u_{t+1} = \frac{1}{|\mathcal{V}_{\mathcal{B}}|} \sum_{u \in \mathcal{V}_{\mathcal{B}}} r^u_{t+1}$. For equation (2b), we let the immediate reward $R(\boldsymbol{s}, \boldsymbol{a})$ equal to the utility $\mathcal{U}_t(\boldsymbol{x})$ at time t. Thus, we have the return $\mathcal{R}(\pi) = \sum_{t=1}^{T} \mathcal{U}_t(\boldsymbol{x})$. Since the (1) is separable about the network agents, we can write the reward of agent $u \in \mathcal{V}_{\mathcal{B}}$ as following.

$$r^u_t = \sum_{l \in \mathcal{V}_{\mathcal{B}}(u)} \sum_{c \in \mathcal{V}_{c,l}(u)} \left[\frac{1}{N_c} U(b_c x_c(t)) - \beta w_l x_c(t) \right] \tag{3}$$

where N_c is the number of agents in the route from *Provider* to *Consumer* c with AGM, $\mathcal{V}_{\mathcal{B}}(u)$ is the set of virtual links connected with to agent u, and $\mathcal{V}_{c,l}(u)$ is the set of *Consumers* that its flow is transcoded or transmitted by agent u. The equation (2c) is based on a standard regularity assumption [32] and holds when for any π, the Markov chain is irreducible and aperiodic [33]. We have the state-value function $V_\pi(\boldsymbol{s})$ and the action-value function $Q_\pi(\boldsymbol{s}, \boldsymbol{a})$ in the following equations.

$$Q_\pi(\boldsymbol{s}, \boldsymbol{a}) = \sum_t \mathbb{E}[\bar{r}_{t+1} - \mathcal{R}(\pi) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi] \tag{4a}$$

$$V_\pi(\boldsymbol{s}) = \sum_{a \in A} \pi(\boldsymbol{s}, \boldsymbol{a}) Q_\pi(\boldsymbol{s}, \boldsymbol{a}) \tag{4b}$$

where $\bar{r}_{t+1} = \frac{1}{|\mathcal{V}_{\mathcal{B}}|} \sum_{u \in \mathcal{V}_{\mathcal{B}}} r^u_{t+1}$. Since the expected long-term average reward is related to problem (1), we can obtain the joint action-value $Q_\pi(\boldsymbol{s}, \boldsymbol{a})$ with equation (1a) based on the joint state $\boldsymbol{s}$ and action $\boldsymbol{a}$ information. To estimate the optimal policy for each agents, we represent the joint policy $\boldsymbol{\pi} = \{\pi^1, \pi^2, ...\}$ to a parametrized function $\boldsymbol{\pi_\theta} = \{\pi^1_{\theta^1}, \pi^2_{\theta^2}, ...\}$.

2) Actor-critic algorithm: the AC algorithm is a method for the maximization reward problem by solving the optimal policy π_θ. According to [33], [34], we have the gradient of the return $\mathcal{R}(\pi_\theta)$ with respect to the policy parameter θ as.

$$\nabla_\theta \mathcal{R}(\theta) = \mathbb{E}_{s \sim d_\theta, a \sim \pi_\theta} \{ \nabla_\theta \log \pi_\theta(\boldsymbol{s}, \boldsymbol{a}) \cdot \mathcal{A}_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a}) \} \tag{5}$$

where $\mathcal{A}_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a})$ is the advantage function which is equal to $Q_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a}) - V_{\pi_\theta}(\boldsymbol{s})$. Thus, we have the sample of the advantage function.

$$\mathcal{A}_t = Q(\boldsymbol{s}_t, \boldsymbol{a}_t) - \sum_{\boldsymbol{a} \in A} \pi_{\theta_t}(\boldsymbol{s}_t, \boldsymbol{a}) Q(\boldsymbol{s}_t, \boldsymbol{a}) \tag{6}$$

For simplicity, we define the sample of $\nabla_\theta \log \pi_\theta(\boldsymbol{s}, \boldsymbol{a})$ as $\boldsymbol{\eta}_t = \nabla_\theta \log \pi_{\theta_t}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The parameter approximation of joint policy $\boldsymbol{\pi}_\theta$ has the following common equation [34].

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \rho_{\theta,t} \cdot \mathcal{A}_t \cdot \boldsymbol{\eta}_t \tag{7}$$

where $\rho_{\theta,t} > 0$ is the stepsize. In our problem, since the reward function is determinate, we only need all agents reach a consensus on the action-value function in critic step. Besides, we need consider the actor step which is responsible for the parameter update (7) of the joint policy.

B. Networked Multi-Agent Reinforcement Learning

When employing networked MARL framework to solve problem (1), we need to consider the network topology of the CLS system. We introduce Networked Multi-Agent MDP [33] which can be represented by a 6-tuple $\mathcal{M} = (\mathcal{V}_{\mathcal{B}}, S, A, P, R, \mathcal{G}(\mathcal{V}, \mathcal{E}))$ with the extension of Dec-MDP, where $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is the network topology. In our design, since a network node is associated with multiple virtual nodes, we consider each node contains multiple agents. We also let the network nodes share the information of local reward with its

neighboring nodes at each time-slot. In MARL, the joint state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$ are globally observable [33], [34]. Besides, the agents $u \in \mathcal{V}_\mathcal{B}$ in CLS has its local action policy $\pi_\theta^u(\boldsymbol{s}, a^u)$ and reward function $R^u(\boldsymbol{s}_t, a_t^u)$.

To facilitate understanding, we first give a brief introduction of the MARL process. At each time-slot t, each agent $u \in \mathcal{V}_\mathcal{B}$ execute a_t^u based on the global state $\boldsymbol{s}_t = \{c_l(t)\}, \forall l \in \mathcal{V}_\mathcal{B}$. As a result, agents in node v get a reward $r_{t+1}^u, u \in \mathcal{V}_\mathcal{B}(v)$ based on the equation (3). After all agents make the joint action $\boldsymbol{a}_t$, the CLS system shifts to a new state $\boldsymbol{s}_{t+1}$ based on the transition probability function $P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$. In addition, the nodes update their parameters based on the reward and share the reward with their neighbors. Note that the agent u make its own action decision a_t^u individually. This means that each agent u has its own policy $\pi_\theta^u(\boldsymbol{s}, a^u)$ which is based on the state $\boldsymbol{s}$. Thus, we have the joint policy $\boldsymbol{\pi}_\theta =< \pi_\theta^u(\boldsymbol{s}, a^u) >_{u \in \mathcal{V}_\mathcal{B}}$.

We focus on the policy $\pi_\theta^u(\boldsymbol{s}, a^u)$ of each agent u. For our problem (1), the function of each agent is to route video flow from *Provider* to *Consumer* on AGM. When an agent receives a flow, it needs to send the flow to one of the next hop agents according to the current state $\boldsymbol{s}_t$. Therefore, $\pi_\theta^u(\boldsymbol{s}, a^u)$ is the probability that agent u selects action a_t^u at state $\boldsymbol{s}_t$. Since $\boldsymbol{s}_t$ contains the available resource of each link $l \in \mathcal{V}_\mathcal{B}$, we restrict that the agent u can only send flow c to the next-hop agent v with sufficient link capacity $c_{(u,v)} \geq w_{(u,v)} x_c(t)$. Moreover, to avoid a loop route, we let the header of flow packet carry a "Nonce" identifier. When an agent detects duplicate "Nonce" in a received packet within one slot, it drops the packet. In this case, the agents u associated with the discarded packet receives the negative immediate reward $r_t^u = -\sum_{l \in \mathcal{V}_\mathcal{B}(u)} \sum_{c \in \mathcal{V}_{c,l}(u)} \beta w_l x_c(t)$. Since we consider a large-scale CLS, the joint state space S is very big. Using a parameterized function to approximate the policy function $\pi_\theta^u(\boldsymbol{s}, a^u)$ is efficient [34]. Because each agent has its own policy, we define the approximate parameters of the agent u as $\theta^u \in \Theta^u$. Further, we define the parameters of policy $\boldsymbol{\pi}_\theta$ as $\theta = [(\theta^1)^\top, ..., (\theta^{|\mathcal{E}_\mathcal{B}|})^\top]^\top \in \Theta$, where $\Theta =< \Theta^u >_{u \in \mathcal{V}_\mathcal{B}}$. To use an AC algorithm with function approximation, we need to make the following standard regularity assumption [32]–[34].

Assumption 1. *For any $u \in \mathcal{V}_\mathcal{B}$, $\boldsymbol{s} \in S$, and $a^u \in A$, the parameterized function $\pi_{\theta^u}^u(\boldsymbol{s}, a^u)$ is always greater than zero for any $\theta^u \in \Theta^u$. Besides, with respect to $\theta^u \in \Theta^u$, $\pi_{\theta^u}^u(\boldsymbol{s}, a^u)$ is continuously differentiable. We define the transition matrix of the joint state $\boldsymbol{s}$ with policy π_θ as P^θ and have:*

$$P^\theta(\boldsymbol{s}'|\boldsymbol{s}) = \sum_{\boldsymbol{a} \in A} \pi_\theta(\boldsymbol{s}, \boldsymbol{a}) \cdot P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}), \quad \forall \boldsymbol{s}, \boldsymbol{s}' \in S \tag{8}$$

Also, we suppose that the evolution of joint state $\{\boldsymbol{s}_t\}_{t \geq 0}$ is irreducible and aperiodic with any π_θ.

It is easy to prove that the above assumptions are true in our problem. In AGM, when the CLS system has no external interference (*e. g.* cross-traffic), the transition probability function P^θ is determinate. This is because when the routing (joint action) $\boldsymbol{a}_t$ and the current system status (joint state) $\boldsymbol{s}_t$ are determined, the next-slot network status $\boldsymbol{s}_{t+1}$ is also determined under no interference. Under interference conditions, since the network environment is uncertain and stochastic [37], the interference can be regarded as Gaussian noise, thus we consider that **Assumption 1** holds.

Next, we consider the goal for the problem (1) in networked MARL framework. We want to design an algorithm that allows all agents to find the optimal joint policy π_θ in a fully cooperative manner. We consider the optimal joint policy π_θ that maximizes the return of the entire CLS system $\mathcal{R}(\pi)$. As defined above, we have the return $\mathcal{R}(\pi) = \sum_{t=1}^T \mathcal{U}_t(\boldsymbol{x})$ and reformulate problem (1) as follows.

$$\max_\theta \ \mathcal{R}(\pi_\theta) = \frac{1}{T|\mathcal{V}_\mathcal{B}|} \lim_{T \to \infty} \sum_{t=1}^T \sum_{c \in \mathcal{V}_\mathcal{B}} \mathbb{E}(r_t^u) \tag{9a}$$

$$\text{s.t. (1b) \& (1c) \& (1d)} \tag{9b}$$

according to (4a) and (4b), we have the following parametrized forms of action-value function and state-value function.

$$Q_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a}) = \sum_t \mathbb{E}[\bar{r}_{t+1} - R(\pi_\theta) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi_\theta] \tag{10a}$$

$$V_{\pi_\theta}(\boldsymbol{s}) = \sum_{a \in A} \pi_\theta(\boldsymbol{s}, \boldsymbol{a}) Q_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a}) \tag{10b}$$

IV. Multi-Agent Actor-Critic Algorithm

In this section, we first introduce the policy gradient theorem for MARL. Then, we design the Multi-Agent Actor-Critic (MAAC) algorithm to solve problem (9) and describe the implementation of the algorithm.

A. Policy Gradient Theorem

One of the main issues for the algorithm design is whether agent u can obtain an approximate parametrized policy $\pi_{\theta^u}^u$ with local information. For this problem, we first need to introduce a policy gradient theorem of networked MARL for our problem as shown in **Theorem 1** which shares a similar proof in based on the single-agent reinforcement learning [36].

Theorem 1. *Based on equations* (2c) *and* (9a)*, we have the return $\mathcal{R}(\pi_\theta) = \sum_{\boldsymbol{s} \in S, \boldsymbol{a} \in A} d_{\pi_\theta}(s) \pi_\theta(\boldsymbol{s}, \boldsymbol{a}) \bar{R}(\boldsymbol{s}, \boldsymbol{a})$ where π_θ is the parametrized policy and $\theta \in \Theta$. Besides, according to* (10a) *and* (6)*, for any agent $u \in \mathcal{V}_\mathcal{B}$, we further have the local parametrized advantage function $\mathcal{A}_{\pi_\theta}^u : S \times A \to R$.*

$$\mathcal{A}_{\pi_\theta}^u(\boldsymbol{s}, \boldsymbol{a}) = Q_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a}) - \widehat{V}_{\pi_\theta}^u(\boldsymbol{s}, \boldsymbol{a}^{\not\in u}) \tag{11}$$

where $\boldsymbol{a}^{\not\in u}$ is the joint action except agent u and $\widehat{V}_{\pi_\theta}^u(\boldsymbol{s}, \boldsymbol{a}^{\not\in u})$ is equal to $\sum_{a^u \in A^u} \pi_{\theta^u}^u(\boldsymbol{s}, a^u) Q_{\pi_\theta}(\boldsymbol{s}, a^u, \boldsymbol{a}^{\not\in u})$. $Q_{\pi_\theta}(\boldsymbol{s}, a^u, \boldsymbol{a}^{\not\in u})$ represents the action-value function of agent u under the condition $\boldsymbol{a}^{\not\in u}$. Since parameter matrix $\theta = [(\theta^1)^\top, ..., (\theta^1)^\top]^\top$, we have the gradient of the return $\mathcal{R}(\pi_\theta)$ with respect to θ^u as following.

$$\begin{aligned} \nabla_{\theta^u} \mathcal{R}(\pi_\theta) &= \mathbb{E}_{s \sim d_\theta, a \sim \pi_\theta} \{\nabla_{\theta^u} \log \pi_{\theta^u}^u(\boldsymbol{s}, a^u) \cdot \mathcal{A}_{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a})\} \\ &= \mathbb{E}_{s \sim d_\theta, a \sim \pi_\theta} \{\nabla_{\theta^u} \log \pi_{\theta^u}^u(\boldsymbol{s}, a^u) \cdot \mathcal{A}_{\pi_\theta}^u(\boldsymbol{s}, \boldsymbol{a})\} \end{aligned} \tag{12}$$

For simplicity, we denote $\phi_{\theta^u}(\boldsymbol{s}, a^u) = \nabla_{\theta^u} \log \pi_{\theta^u}^u(\boldsymbol{s}, a^u)$ as the individual score function for agents u.

Proof. Based on [36], we give the proof of **Theorem 1**. We have the joint policy $\boldsymbol{\pi}_\theta =< \pi_\theta^u >_{u \in \mathcal{V}_\mathcal{B}}$. Since each agent

make action decision independently, we consider that $\boldsymbol{\pi}_\theta = \prod_{u\in\mathcal{V}_\mathcal{B}} \pi^u_\theta$. According to the gradient of the return $\mathcal{R}(\pi_\theta)$ with respect to θ in single-agent reinforcement learning [36], we have the following equation

$$\begin{aligned}\nabla_\theta \mathcal{R}(\pi_\theta) &= \mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\{\nabla_\theta \log \pi_\theta(\boldsymbol{s},\boldsymbol{a})\cdot \mathcal{Q}_{\pi_\theta}(\boldsymbol{s},\boldsymbol{a})\} \\ &= \sum_{\boldsymbol{s}\in S} d_\theta \sum_{\boldsymbol{a}\in A}\pi_\theta \left[\sum_{u\in\mathcal{V}_\mathcal{B}} \phi_{\theta^u}(\boldsymbol{s},a^u)\right]\cdot \mathcal{Q}_{\pi_\theta}(\boldsymbol{s},\boldsymbol{a})\}\end{aligned} \tag{13}$$

where Q_{π_θ} is the stationary distribution with policy π_θ. Since the policy of different agents is independent, we further have the gradient of $\mathcal{R}(\pi_\theta)$ with respect to parameter θ^u as

$$\nabla_{\theta^u}\mathcal{R}(\pi_\theta) = \sum_{\boldsymbol{s}\in S} d_\theta \sum_{\boldsymbol{a}\in A}\pi_\theta \phi_{\theta^u}(\boldsymbol{s},a^u)\mathcal{Q}_{\pi_\theta}(\boldsymbol{s},\boldsymbol{a}) \tag{14}$$

Moreover, consider the following equation.

$$\begin{aligned}&\sum_{\boldsymbol{a}\in A}\pi_\theta\phi_{\theta^u}(\boldsymbol{s},a^u)\widehat{V}^u_{\pi_\theta}(\boldsymbol{s},\boldsymbol{a}^{\not\in u}) \\ &= \sum_{\boldsymbol{a}^{\not\in u}\in A^{\not\in u}}\left[\widehat{V}^u_{\pi_\theta}(\boldsymbol{s},\boldsymbol{a}^{\not\in u})\prod_{v\neq u}\pi^v_\theta\right]\left[\sum_{a^u\in A^u}\pi^u_{\theta^u}\phi_{\theta^u}(\boldsymbol{s},a^u)\right]\end{aligned} \tag{15}$$

where $A^{\not\in u}$ is the joint action space without agent u. We discuss $\sum_{a^u\in A^u}\pi^u_{\theta^u}\phi_{\theta^u}(\boldsymbol{s},a^u)$ which is equal to $\sum_{a^u\in A^u}\pi^u_{\theta^u}\nabla_{\theta^u}\log\pi^u_{\theta^u}(\boldsymbol{s},a^u)$. Taking the derivative of this formula we have $\sum_{a^u\in A^u}\nabla_{\theta^u}\pi^u_{\theta^u}(\boldsymbol{s},a^u)$. Since $\sum_{a^u\in A^u}\pi^u_{\theta^u}(\boldsymbol{s},a^u)$ is always equal to 1, we have

$$\nabla_{\theta^u}\sum_{a^u\in A^u}\pi^u_{\theta^u}(\boldsymbol{s},a^u) = 0 \tag{16}$$

Thus, the equation (15) is equal to 0. **Theorem 1** is proved. □

From **Theorem 1**, we know that if each agent has an unbiased estimate of the global action-value function $Q(\boldsymbol{s},\boldsymbol{a})$ or advantage function $A(\boldsymbol{s},\boldsymbol{a})$, they can calculate the policy gradient with score function $\phi_{\theta^u}(\boldsymbol{s},a^u)$. This motivates us to design an algorithm that can get consensus reward and estimate policy function well for each agent. Next, we will focus on the proposed MAAC algorithm and its implementation.

B. Multi-Agent Actor-Critic Algorithm

According to (10a) and (11), the unbiased estimate of action-value and advantage functions require global reward $\{r^u_t\}_{u\in\mathcal{V}_\mathcal{B}}$ information. However, the distributed agents only have their local reward function r^u_t which may lead to inaccurate estimation. To solve the problem, we let each node v share its agents' reward with its neighbors on the network. We denote the sharing reward of node v at time t as $\widehat{r}^v_t$. In this way, the agents can reach a consensual estimate of action-value function which further improves the policy of each agent [34]. The MAAC algorithm is stated as **Algorithm 1** and we describe it from the agent $u\in\mathcal{V}_\mathcal{B}$ perspective next.

As the algorithm shows, we set the non-negative stepsize β_ϕ and β_θ and the initial parameter θ^u_t of the parameterized function π_{θ^u} for each agent u. The architecture of the proposed solution is illustrated in Fig. 3. In every time slot t, the

Algorithm 1: Multi-agent Actor-critic Algorithm

Input: the nonnegative stepsizes β_θ and β_ϕ; random $\theta^u_t, \forall u\in\mathcal{V}_\mathcal{B}$; initial state $\boldsymbol{s}_0$ of the system; initial policy $\pi^u_{\theta^u_0}(\boldsymbol{s}_0,a^u)$ and action a^u_0, $u\in\mathcal{V}_\mathcal{B}$.

while $t\in\mathcal{T}$ **do**
  foreach *Agent* $u\in\mathcal{V}_\mathcal{B}$ **do**
    obtain the current joint state $\boldsymbol{s}_t$
    calculate the local reward $\widehat{r}^u_t$ based on (3);
    /*get routing action based on local policy*/
    $a^u_{t+1}\leftarrow\pi^u_{\theta^u_t}(\boldsymbol{s}_t,\cdot)$ and execute routing action;
    /*complete the routing*/
    the last-hop agent inform the on-path agents whether the routing is successful;
  end
  foreach *Node* $v\in\mathcal{V}$ **do**
    /*calculate the reward of node*/
    $\widehat{r}^v_t\leftarrow\frac{1}{|\mathcal{V}(v)|}\sum_{u\in\mathcal{V}(v)}\widehat{r}^u_t$, where $u\in\mathcal{V}(v)$;
    share $\widehat{r}^v_t$ with one-hop neighbors based on the network topology $\mathcal{G}(\mathcal{V},\mathcal{E})$;
    /*consensus step*/
    $r^v_{t+1}\leftarrow\frac{1}{|Nei(v)|}\sum_{i\in Nei(v)}\widehat{r}^i_t$
    /*where $|Nei(v)|$ is the number of neighbors*/
  end
  foreach *Agent* $u\in\mathcal{V}_\mathcal{B}$ **do**
    /*critic step*/
    $Q_t\leftarrow Q_{t-1}+\mathbb{E}[r^v_t-\frac{1}{t}\sum_{\tau=1}^t r^v_\tau]$;
    $\phi^u_t\leftarrow\nabla_{\theta^u}\log\pi^u_{\theta^u_t}(\boldsymbol{s}_t,a^u_t)$;
    $A^u_t\leftarrow Q_t-\sum_{a^u\in A^u}\pi^u_{\theta^u_t}(\boldsymbol{s}_t,a^u,\boldsymbol{a}^{\not\in u},\theta^u_t)$;
    /*actor step*/
    $\theta^u_{t+1}\leftarrow\theta^u_t+\beta_\theta\cdot A^u_t\cdot\phi^u_t$;
  end
end

Consumers will request specific live videos of CLS based on their preferences. In our algorithm, each agent first obtains the current CLS system states (the joint states) $\boldsymbol{s}_t$ from a specific central CS. Based on equation (3), the agent calculates the local reward $\widehat{r}^u_t$. Next, agent u obtains the local action a^u_{t+1} according to the local policy $\pi^u_{\theta^u}(\boldsymbol{s}_t,\cdot)$ and executes the flow routing. When routing is completed, the last-hop agents shares the information of whether the routing is successful with all agents over the routing path. When routing fails (flow cannot reach the *Consumer*), such as in a loop path, the nodes on the routing will get a negative reward. Next, each network node will calculate the average reward of its agents $\widehat{r}^v_t=\frac{1}{|\mathcal{V}(v)|}\sum_{i\in\mathcal{V}(v)}\widehat{r}^i_t$ which will be further shared with its one-hop neighbors through the communication network $\mathcal{G}(\mathcal{V},\mathcal{E})$. In order to reach a consensual reward, each network node v will execute the **consensus step** by averaging the neighbors' reward $\widehat{r}^i_t,\forall i\in Nei(v)$ where $Nei(v)$ is the set of neighbor nodes for network node v. For each agent $u\in\mathcal{V}_\mathcal{B}$, in **critic step**, it calculate the action-value Q_t based on the average reward of its neighbors r^v_t and previous action-value Q_{t-1}. Then, the agents u update their score function value ϕ^u_t and advantage function value A^u_t based on the joint

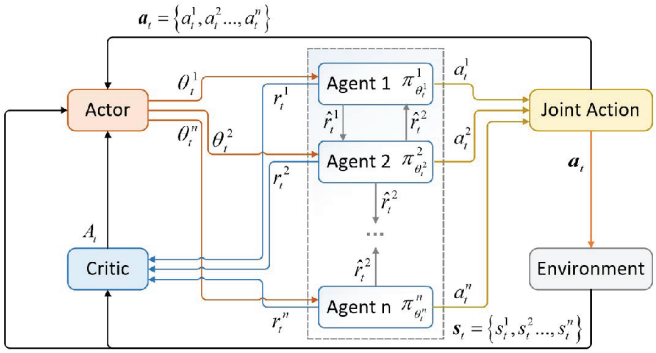

Fig. 3. The architecture of the proposed algorithm

states $\boldsymbol{s}_t$, joint actions $\boldsymbol{a}_t$ and local policy $\pi^u_{\theta^u_t}$. Note that, $\pi^u_{\theta^u_t}(\boldsymbol{s}_t, a^u, \boldsymbol{a}^{\not\subseteq u}, \theta^u_t)$ represents the action-value function with the condition $\boldsymbol{a}^{\not\subseteq u}$ and local policy $\pi^u_{\theta^u_t}$. After that, the agents execute the **Actor step** and update the policy parameter θ^u_{t+1} based on the values of score function and advantage function.

Distributed Implementation: The MAAC algorithm consists of three phases. The first phase is the routing action selection of each agents which needs the global system state $\boldsymbol{s}$ from a central server and local policy $\pi^u_{\theta^u}$. For the second phase, the network nodes only require one-hop reward information exchange to reach a consensus of global reward. In addition, for each agent u in phase three, it needs the joint states $\boldsymbol{s}$ and joint actions $\boldsymbol{b}$ to calculate the value of score function and advantage function respectively. Since the agents obtain the joint states in the first phase, the agents only need exchanging information with the nodes on the routing path to get the joint action. Although the state information needs to be maintained by a central server, the reward function and the policy are local and can be considered as a decentralized solution in MARL [25]–[27], [34], [35]. Hence, the MAAC algorithm can be deployed in a distributed manner with only joint states $\boldsymbol{s}$ and joint actions $\boldsymbol{a}$ information sharing. The space complexity and time complexity of MAAC are $\mathcal{O}(\mathcal{E}_\mathcal{B})$ and $\mathcal{O}(\mathcal{V}_\mathcal{B})$ where $|\mathcal{E}_\mathcal{B}|$ and $|\mathcal{V}_\mathcal{B}|$ are the number of virtual links and the number of virtual nodes respectively. This shows that the complexity of the algorithm increases linearly with the scale of CLS system, which is scalable. To prove the feasibility of the MAAC algorithm, we further evaluate the bandwidth cost of signalling and compare it with those of two state-of-the-art solutions [11], [16] in Section V.

V. Performance Evaluation

In this section, we introduce the experimental scenario and parameter settings. Then we use single-agent reinforcement learning as benchmark and analyse the numerical results of our proposed decentralized algorithm. Finally, we evaluate the performance of our solution in a prototype system and compare it with **E**dge-**C**loud (EC) cooperative solutions [11] and **C**loud-**C**rowd (2C) solutions [16].

A. Experiment Setup

We conduct a series of numerical simulations to validate the convergence and optimality of the MAAC algorithm and realise a prototype system based on an open-source framework

TABLE I
BANDWIDTH REQUIREMENT AND TRANSCODING COST

resolution	1080p 60fps	1080p	720p 60fps	720p	480p	360p
bandwidth (Mbps)	5.86	4.45	2.75	1.93	1.10	0.52
transcoding (vCPU use)	454%	333%	210%	142%	81.6%	50.5%

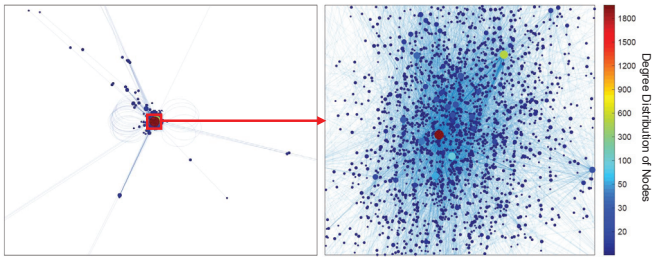

Fig. 4. The topology diagram of the large-scale CLS system

srs [38] for further validation of service performance. To better evaluate the effectiveness of our solution, we use a real-world dataset [39] that was crawled from the official APIs [40] to reconstruct a more realistic evaluation environment. The dataset contains trace records of more than 1.5 million broadcasters and 9 million streams from Feb. 1st to 28th, 2015. Each trace records every 5 mins and includes stream ID, source resolution, stream start/end time, the number of viewers, and so on. Since most broadcasters with too few viewers often have small variations in the number of their audience, we selected the stream traces with more than 1000 concurrent viewers as driving data to generate requests for our numerical simulation.

To evaluate the numerical performance of our MAAC algorithm, we used Inet [41], an open-source communication network generators, to generate a fully connected Internet topology with 5000 nodes, as shown in Fig. 4. We set the node with the highest degree (1922) as the central server and the nodes with a degree of over 50 as CSs. In addition, we set as ESs the nodes with degrees greater than 20 and less than 50 and set about 4390 nodes with degrees less than 20 as amateur broadcasters and viewers. CSs are connected with each other by 300Mbps links and are connected to ESs via 100Mbps links. The viewers and broadcasters connect to the servers (CSs or ESs) through a link with bandwidth ranging from 1Mbps to 10Mbps. The bandwidth of links between the viewers is set to 10Mbps. We consider that the ESs have sufficient computing resources, the maximum computing resources unit of ESs is 20, and the viewers' resource is set to 1. We set the number of CLS channels to 100, and each channel has six video resolutions. According to [11], we set up the requirements of bandwidth and transcoding cost of different resolution as shown in Table 1, which are measured by Twitch's official video statistic tool and AWS c4.8xlarge instance, respectively.

The initial available capacities of bandwidth $c_{(u,v)}$ and computing c_u for ESs and viewers are uniformly sampled in the range $(0, c_{max}]$. Besides, we set the parameters as follows: $\beta_\theta = \beta_\phi = 0.01$, $\beta = 0.5$, $\theta^u_0 \in (0,1)$ with uniform distribution and $w_{(u,v)} = 3w_u = 3, \forall u, v \in \mathcal{V}$. We

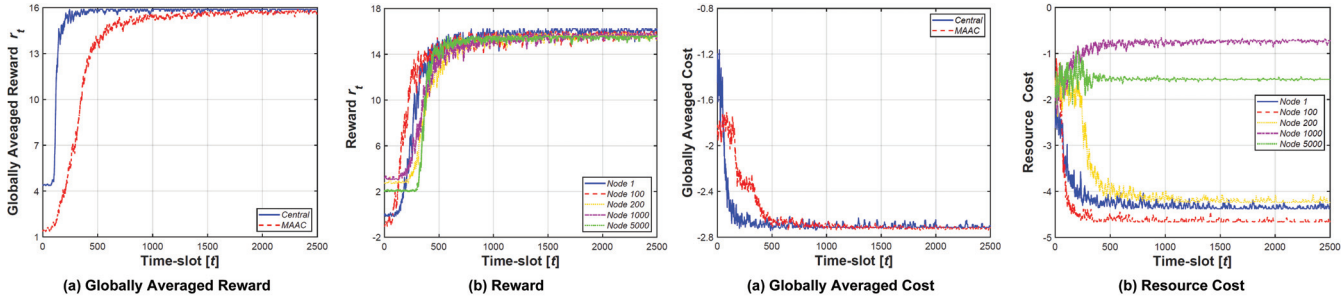

Fig. 5. Comparison of main theoretical results. (a) The optimality and convergence of globally averaged reward; (b) The convergence of local averaged reward of different agents (1, 100, 200, 1000, 5000); (c) The convergence of globally averaged resources cost; (d) the resources cost of different agents

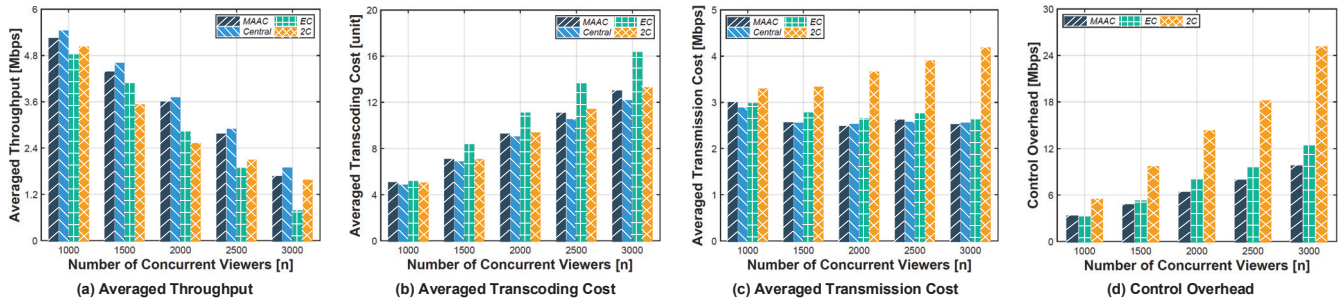

Fig. 6. Comparison of (a) averaged throughput, (b) averaged transcoding cost, (c) averaged transmission cost and (d) signalling overhead of control message with three (**Central**, **EC**, **2C**) solutions.

let the parameterized function of policy $\pi^u_{\theta^u}(\boldsymbol{s}, a^u)$ follow the Boltzman policy.

$$\pi^u_{\theta^u}(\boldsymbol{s}, a^u) = \frac{\exp\left(\lambda^\top_{a^u}\theta^u\right)}{\sum_{b^u \in A^u}\exp\left(\lambda^\top_{b^u}\theta^u\right)} \tag{17}$$

where $\lambda^\top_{a^u}$ is the feature vector for a specific action of agent u. The dimension of the feature vector is the same as θ^u, and both are related to the dimension of the action space for agent u. Thus, we have the form of the score function for agent u.

$$\phi_{\theta^u}(\boldsymbol{s}, a^u) = \lambda_{\boldsymbol{s},a^u} - \sum_{b^u \in A^u} \pi^u_{\theta^u}(\boldsymbol{s}, a^u)\lambda_{\boldsymbol{s},b^u} \tag{18}$$

B. Evaluation Results

We consider a centralized reinforcement learning approach with global information as baseline and further compare the performance of throughput and resource-saving with two state-of-the-art solutions in a large-scale CLS system. **Central** approach is the benchmark, which has only one agent with global reward and policy functions. In each time-slot, the agent obtains a joint action (route) based on system state and global policy function, and then update the parameters θ of the policy function according to the global reward. **EC** solution [11] employs a novel deep neural network based algorithm to maximize the personalized QoE by selecting an efficient transcoding-delivery path over cloud-edge infrastructure for each viewer. **2C** solution [11] leverages ubiquitous crowd devices to assist the online video transcoding. It lets the regional data center allocate the transcoding tasks to the crowd and collect the transcoded data from viewers.

As shown in Fig. 5, we first conduct some numerical studies to verify the optimality and convergence of the MAAC algorithm and compare it with **Central**. We see from Fig. 5 (a) that, both solutions converge to the optimal averaged reward within around 250 and 1000 time-slots, respectively. This shows that our method can achieve centralized performance with an acceptable expense of convergence. Fig. 5 (b) presents the reward convergence of different agents (1, 100, 200, 1000, 5000). We can see that all the agents converge to the same value which is equal to the optimal reward. This proves that the consensus scheme is convergent and can provide an unbiased estimate of global reward. Next, we analyze the global averaged resources cost and individual resource cost of agents. Note that, since the agents play different roles (*e.g. Consumer*, *Transcoder*) in the CLS system, the time-slot to reach the stable phase varies for different nodes. Among them, nodes 1, 100 and 200 are the CSs or ESs which are often used as *Transcoders*, so they consume more resources. Besides, nodes 1000 and 5000 are Providers and Consumers, respectively so they spend a small number of resources. The results are shown in Fig. 5 (c) and (d), respectively.

We compare the proposed MAAC algorithm with **Central**, **EC** and **2C** solutions on throughput, resources saving and control overhead in different system scale. In Fig. 6 (a), as the number of concurrent viewers grows, the averaged throughput of the four solutions decreases and our solution has the highest throughput compared to **EC** and **2C**. Fig. 6 (b) and (c) present the averaged transcoding and transmission cost. By leveraging computing resources of crowd, our solution and **2C** have good scalability in terms of transcoding resources cost. In contrast, **EC** only uses resources of CSs and ESs, so as the number of users increases, the averaged transcoding overhead grows faster. Besides, compared with **2C**, our solution has better performance in terms of saving bandwidth. Since **2C** requires the regional data center to frequently deliver content to viewers for cooperative transcoding, this brings a lot of traffic to the CLS system. To prove the feasibility of the MAAC algorithm, we evaluate the overall bandwidth cost of signalling with **EC** and **2C**. In our method, the action information exchange on the delivery path and global state distribution is required. In **EC**, the signalling overhead is mainly caused by the commu-

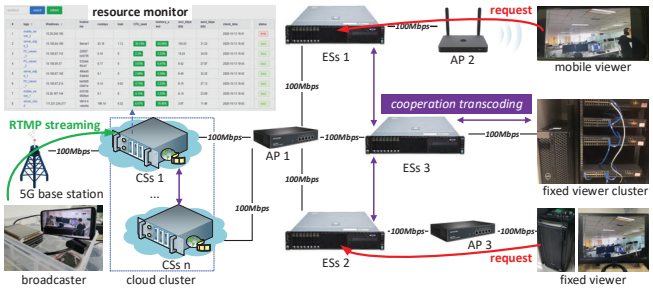

Fig. 7. The topology diagram of the prototype system

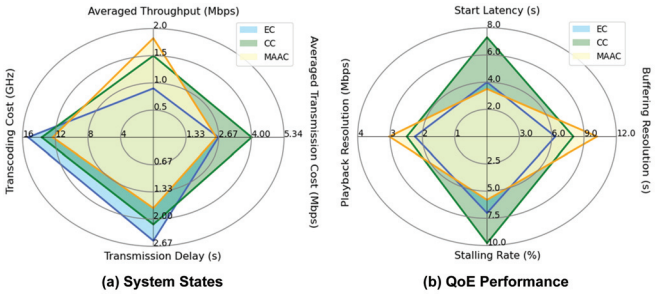

Fig. 8. The experiment results of (a) system states and (b) QoE performance.

nication of workload status between different servers (CSs and ESs) and gathering of playback information of all viewers. In **2C**, the regional data center needs to maintain the viewing state of all crowd devices and allocate the transcoding task, which requires monitoring the viewers' status continuously. Fig 6 (d) shows how our method has the lowest control overhead among the three solutions.

Fig 7 shows the prototype system built. It includes one broadcaster, three CSs, three ESs, one cluster of fixed viewers and two clusters of mobile viewers. We rented ecs.g5.2xlarge (Intel Xeon Platinum 8163, 2.5Ghz eight core, 32GB memory) of aliyun as CSs. Further, we used three servers (Xeon Bronze 3104, 1.7Ghz eight core, 32GB memory) to deploy ESs and set up four virtual machines (Intel Core i7-7700k, 4.2Ghz dual-core, 4GB memory) as the cluster of fixed viewers through VMware. Also, we used the python package **psutil** [42] to get the resource usage of Linux server, including CPU utilization (%) and import/export bandwidth (kbps) of CS, RSs and ESs. All of our experiments were conducted on CentOS 7. We used **yasea** [43] (an RTMP live streaming client for Android) to push the streaming from the broadcaster (HUAWEI Nova 6 5G) to CS through the 5G network. We let the broadcasters upload the content to CSs 1 through Real-Time Messaging Protocol (RTMP). Once the viewers request livecast services, the request is captured by the ESs and then sent to the CSs 1. Further the CSs and ESs route the RTMP flow from CSs 1 to viewer based on their well trained policy $\pi_\theta(s, a)$. In our prototype system, the ESs and CSs used **ffmpeg** [44] to convert the RTMP streaming into multi-resolution. In addition, the **EC** solution first selected a transcoding delivery path, and then dynamically performed allocation of the transcoding task according to the workload of CSs and ESs on the path. In the **2C** solution, the CSs and ESs offloaded the transcoding tasks to the fixed viewers and then collected transcoded content to service the mobile viewers.

From Fig. 8 (a), we can observe four indicators of MAAC **EC** and **2C** based on the prototype-based testing. MAAC maintains the highest averaged throughput and the lowest transmission delay from broadcaster to viewer from the three solutions. In addition, we monitored the CPU usage and input and output bitrate of all servers. The overall transcoding cost of MAAC is about 12.3GHz which saves CPU usage by about 20% and 10% in comparison with **EC** and **2C**, respectively. The average bandwidth cost of MAAC is similar to that of **EC**, but better than that of **2C**. Moreover, we have also analyzed some metrics of QoE in livecast. We considered four indicators including start-up latency, stalling rate, playback resolution and buffering ratio/time. During the experimental period, we let servers request random RTMP streams from each other to simulate network cross-traffic. We conducted an uninterrupted CLS experiment and got the results shown in Fig. 8 (b). We found that our solution can dynamically adjust the transcoding delivery path when the CLS system states changes. Compared with the other solutions, our method reduced the start-up delay by 12.5% and stalling rate by 17%, and provided 20% and 25% improvements in playback bitrate and buffering time.

VI. Conclusion and Future Work

This paper proposed a novel Augmented Graph Model for large-scale CLS systems, which cleverly combines transcoding and delivery by adding virtual nodes and links based on the network topology. With the help of AGM, we transformed the complex joint resource optimization problem of computing and transmission into a network routing problem. We further formalized the network routing problem which jointly considers the resource cost and service performance. To meet scalability requirements of CLS, we proposed an innovative multi-agent reinforcement learning solution which enables each agent in the CLS system has its own local reward function and routing policy. We further designed a distributed multi-agent actor-critic algorithm based on policy gradient to overcome the drawbacks of Q-learning as mentioned above. Our algorithm achieves only linear complexity with respect to the increase in the number of virtual nodes and links. Thorough simulation and prototype system-based testing have verified the performance of our solution in comparison with state-of-the-art alternative approaches. Compared with two mainstream solutions, the results show that our approach provides a significant improvement in terms of both saving resources and improving service performance. Future work will address the challenge and deploy our algorithm on a CLS platform to verify its commercial benefit in a real environment. It is hoped that our solution can efficiently serve a large CLS system and enable high-quality live video services.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant 61871048, Grant 61872253, and Grant 62001057; by the BUPT Excellent Ph.D. Students Foundation CX2019205; by National Key R&D Program of China (2018YFE0205502). G.-M. Muntean acknowledges the Science Foundation Ireland Research Centres Programme support under Grant Nos. SFI/12/RC/2289 P2 (Insight Centre for Data Analytics) and 16/SP/3804 (ENABLE).

REFERENCES

[1] Twitch, *https://www.twitch.tv/*.
[2] Douyu.tv, *https://www.douyu.com/*.
[3] TwitchTracker, *https://twitchtracker.com/statistics*.
[4] TouBang Statistics, *http://www.toubang.tv/data/platform*.
[5] Douyu Annual Reports, *https://ir.douyu.com/Annual-Reports*.
[6] Y. Zheng, *et al.*, "Online Cloud Transcoding and Distribution for Crowdsourced Live Game Video Streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1777-1789, Aug. 2017.
[7] C. Dong, *et al.*, "Joint Optimization of Data-Center Selection and Video-Streaming Distribution for Crowdsourced Live Streaming in a Geo-Distributed Cloud Platform," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 729-742, June 2019.
[8] R. Aparicio-Pardo, *et al.*, "Transcoding live adaptive video streams at a massive scale in the cloud," *ACM Multimedia Systems Conference (MMSys '15)*, ACM, New York, NY, USA, pp. 49-60, 2015.
[9] C. Dong, *et al.*, "A Novel distribution service Policy for Crowdsourced Live Streaming in Cloud Platform," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 679-692, June 2018.
[10] M. Ma, *et al.*, "Characterizing User Behaviors in Mobile Personal Livecast: Towards an Edge Computing-assisted Paradigm," *ACM Trans. Multimedia Comput. Commun. Appl.* vol. 14, no. 66, July 2018.
[11] H. Pang, *et al.*, "Optimizing Personalized Interaction Experience in Crowd-Interactive Livecast: A Cloud-Edge Approach," *ACM Conference on Multimedia (MM '18)*, ACM, New York, NY, USA, 2018.
[12] Z. Wang, *et al.*, "A Joint Online Transcoding and Delivery Approach for Dynamic Adaptive Streaming," *IEEE Transactions on Multimedia*, vol. 17, no. 6, pp. 867-879, June 2015.
[13] Q. He, *et al.*, "Fog-Based Transcoding for Crowdsourced Video Livecast," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 28-33, April 2017.
[14] Y. Zhu, *et al.*, "When Cloud Meets Uncertain Crowd: An Auction Approach for Crowdsourced Livecast Trans-coding," *ACM Conference on Multimedia (MM '17)*, ACM, New York, NY, USA, 2017.
[15] Q. He, *et al.*, "CrowdTranscoding: Online Video Transcoding With Massive Viewers," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1365-1375, June 2017.
[16] Y. Zhu, *et al.*, "When Crowd Meets Big Video Data: Cloud-Edge Collaborative Transcoding for Personal Livecast," *IEEE Trans. on Net. Sci. and Engin.*, vol. 7, no. 1, pp. 42-53, 1 Jan.-March 2020.
[17] X. Chen, *et al.*, "Augmented Queue-based Transmission and Transcoding Optimization for Livecast Services Based on Cloud-Edge-Crowd Integration," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1-1, Dec. 2020.
[18] A. Sinha and E. Modiano, "Optimal control for generalized network-flow problems," *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, Atlanta, GA, 2017, pp. 1-9.
[19] J. Zhang, *et al.*, "Optimal Control of Distributed Computing Networks with Mixed-Cast Traffic Flows," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, 2018, pp. 1880-1888.
[20] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," *10th International Conference Machine Learning (ICML)*, Amherst, OH, Jun. 1993, pp. 330-337.
[21] G. Stampa, *et al.*, "A deep-reinforcement learning approach for software-defined networking routing optimization," *arXiv preprint arXiv:1709.07080*, 2017.
[22] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route with deep rl," *NIPS*, 2017.
[23] P. Sun, *et al.*, "Sinet: Enabling scalable network routing with deep reinforcement learning on partial nodes," *ACM SIGCOMM 2019 Conference*, 2019, pp. 88-89.
[24] G. Knzel, *et al.*, "Latency and Lifetime Enhancements in Industrial Wireless Sensor Networks: A Q-Learning Approach for Graph Routing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5617-5625, Aug. 2020.
[25] X. You, *et al.*, "Toward packet routing with fully-distributed multi-agent deep reinforcement learning," *arXiv preprint arXiv:1905.03494*, 2019.
[26] H. Mao, *et al.*, "Learning multi-agent communication under limited-bandwidth restriction for internet packet routing," *arXiv preprint arXiv:1903.05561*, 2019.
[27] X. Li, *et al.*, "Routing Protocol Design for Underwater Optical Wireless Sensor Networks: A Multi-Agent Reinforcement Learning Approach," *IEEE Internet of Things Journal*, pp. 1-1, April 2020.
[28] R. Lowe, *et al.*, "Multiagent actor-critic for mixed cooperative-competitive environments," *NIPS*, 2017, pp. 20942100.
[29] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing," *IEEE/ACM Transaction Networking*, vol. 24, no. 2, pp. 996-1010, Apr. 2016.
[30] T. Tran, *et al.*, "Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks," *IEEE Transactions on Mobile Computing*, pp. 1-1, Sep. 2018.
[31] M. Wang, *et al.*, "Design of Multipath Transmission Control for Information-Centric Internet of Things: A Distributed Stochastic Optimization Framework," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9475-9488, Dec. 2019.
[32] V. R. Konda, *et al.*, "Actor-critic algorithms," *Advances in Neural Information Processing Systems*, 2000, pp. 10081014.
[33] K. Zhang, *et al.*, "Networked multi-agent reinforcement learning in continuous spaces," *2018 IEEE Conference on Decision and Control (CDC)*, pp. 2771-2776, 2018.
[34] K. Zhang, *et al.*, "Fully decentralized multi-agent reinforcement learning with networked agents," *the 35th International Conference on Machine Learning, ICML*, 2018.
[35] H. Hao, *et al.*, "A Multi-update Deep Reinforcement Learning Algorithm for Edge Computing Service Offloading," *Proceedings of the 28th ACM International Conference on Multimedia*. pp. 3256-3264. Seattle, WA, USA, 2020.
[36] R. Sutton *et al.*, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, 2000.
[37] J. P. Champati, *et al.*, "Transient Analysis for Multihop Wireless Networks Under Static Routing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 722-735, April 2020.
[38] Simple Realtime Server (srs), *https://github.com/ossrs/srs*.
[39] Clivecast, *https://clivecast.github.io/*.
[40] Developer APIs, *http://dev.twitch.tv/*.
[41] Inet Topology Generator, *http://topology.eecs.umich.edu/inet/*.
[42] psutil, *https://pypi.org/project/psutil/*
[43] yasea, *https://github.com/begeekmyfriend/yasea*
[44] ffmpeg, *https://ffmpeg.org/*