Fast Near-Optimal Algorithm for Delivering Multiple Live Video Channels in CDNs

Jiayi Liu and Gwendal Simon

Telecom Bretagne (Institut Mines-Telecom), Rennes, France

La popularité croissante des applications vidéos et le développement de technologies de diffusion de flux à débits variables ont un impact fort sur la charge imposée à l'infrastructure des réseaux de diffusion de contenus (CDN). Dans ce papier, nous abordons le problème de l'utilisation optimale du réseau de diffusion dans le modèle discret de diffusion de flux. Nous formulons le problème générique, puis nous proposons un algorithme rapide et quasi-optimal dans le cas où les équipements du réseau de diffusion ont la même capacité de transmission.

Keywords: Video streaming, CDN, Approximate Algorithm

1 Introduction

Content Delivery Networks (CDNs) should support a sharp increase in live video streaming consumption. In addition, CDNs have to deal with the growing adoption of *rate-adaptive streaming* techniques such as the recent MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard. In DASH, a server can offer multiple *representations* of the same video channel. The server thus provides clients the flexibility of choosing the video representation that fits their device capabilities and their network capacity. DASH conveniently addresses the problem of delivering videos to a heterogeneous population of clients, but DASH imposes the whole set of representations (with an aggregated bit-rate over 30 Mbps) to be sent from the CDN sources to the CDN edge-servers. As a consequence, the traffic within CDN infrastructure explodes [Ing12]. A new challenge is thus to cope with *under-provisioned* CDN infrastructures.

The previous theoretical works related to live streaming in CDNs have highlighted the main characteristics of these networks, in particular the 3-tier topology (origin servers, reflectors and edge-servers) [NSS10], and the restriction on the upload capacity of the equipment [ZAB$^+$12, ASV11]. The goal of these previous works is to reduce the transmission cost of video delivery. However, the development of peering agreements between CDN and network operators has reduced the importance of transmission cost in CDN. In the meantime, the concerns related to the capacity of CDN infrastructure are growing. In this paper, we neglect transmission costs and we focus instead on video delivery in a network where the upload capacity of equipments is limited. This problem is in essence similar as the one that is adressed in a series of works related to the *streaming capacity* of networks [SLC$^+$11, ZLW11], which aim to determine the maximum bit-rate that can be delivered to all nodes in a network. Both problems aim at optimizing the utilization of network resources in the context of data streaming. These works however use an idealized model where data streams are infinitely divisible. In comparison, a CDN has to deliver a set of independent non-divisible data streams, which need to be either delivered in whole, or not delivered at all. The goal is to maximize the number of delivered streams in the network, which we refer to as the *discretized streaming model*.

In this paper, we introduce an optimization problem, which is the generic formulation of the *discretized streaming capacity problem*. The practical goal of this problem is to maximize the number of delivered streams in CDN infrastructure, with regard to some preferences set by the CDN provider and subject to the limited upload capacity of CDN equipments. This problem is NP-complete (due to lack of place, we do not give the proof in this paper). We then present a fast near-optimal algorithm, which applies to a practical case where all intermediate equipments have similar upload capacity. This scenario is motivated by the observation that CDN providers tend to use one customized type of equipments, for example the Netflix

appliance hardware [net12]. In this paper, we develop the theoretical analysis of this algorithm, the practical evaluation being skipped due to lack of room.

2 System model and problem definition

We consider a typical 3-tier CDN topology (*e.g.* Akamai [NSS10]). There are three types of communication devices : a small number of *sources*, a medium size network of *reflectors*, and a large number of *edge servers*. The topology of a CDN is modeled by a directed graph $G = (V,E)$, where V represents the communication devices, and E represents the communication links. Let $V_S, V_R, V_E \subset V$ be the set of sources, reflectors and edge servers, respectively. They are constrained by their uplink capacity $c(v), \forall v \in V$. There are three types of possible connections in E : E_{SR} connects sources to reflectors, E_{RR} allows communication between reflectors, and E_{RE} delivers the various representations of the live stream to the edge servers.

The live streams consist of l different channels. The raw video of each channel is transcoded into k representations, where the bit-rate of the i-th representation, $1 \leq i \leq k$, is λ_i. For simplicity of notation hereafter we denote by $[m]$ the integer interval $\{1,\ldots,m\}$. Also, let d_{ij} be the i-th representation of the j-th channel, $i \in [k]$, $j \in [l]$. Each representation d_{ij}, $i \in [k]$, $j \in [l]$, is delivered through a set of trees in the network. We denote by $\mathcal{T}_{ij}$ the set of delivery trees for d_{ij}.

Ultimately a CDN provider would like every edge server to receive all the representations it requires. This however might not be possible due to the capacity constraint of the CDN infrastructure. In such case, the CDN provider leverages statistics to prioritize the delivery [NSS10]. The preferences of edge servers in respect to the available representations is captured in a *utility score*, such that α_u^{ij} is the utility score that edge server u assigns to representation d_{ij}. To evaluate the performance of a delivery scheme, the idea is thus to evaluate a *utility score function* $\alpha_u(\mathbf{X_u})$ for each edge server $u \in V_E$ as follows :

$$\alpha_u(\mathbf{X_u}) = \sum_{i\in[k]} \sum_{j\in[l]} \alpha_u^{ij} x_u^{ij}$$

where $\mathbf{X_u}$ is an indicator matrix of size $k \times l$ such that x_u^{ij} has a value of 1 if u receives d_{ij} and 0 otherwise.

The *Maximum Average Utility Score* (MAUS) problem is essentially the maximization of the *average* utility score function of the edge servers, as summarized below.

Problem 2.1 (MAUS)**.** *Given the topology and capacity constraints of a CDN, find delivery tree sets,* $\{\mathcal{T}_{ij}\}_{i\in[k],j\in[l]}$*, such that* $\sum_{u\in V_E} \alpha_u(X_u)$ *is maximized.*

Please note that this problem is a general optimization problem. We proved that this problem is NP-complete based on a reduction from 3-SAT. Due to the space limitation, we could not provide the proof here (full paper under revision).

3 A fast near-optimal algorithm for practical scenario

We focus on a practical scenario where all equipments are homogeneous. This scenario is typical from companies that deploy an infrastructure using customized routers. For example, the CDN providers in charge of delivering worldwide popular events (*e.g.*, Olympic Games) develop the infrastructure according to the specific needs of the event and based on a set of homogenous equipments. This homogeneity results in a similar upload capacity, i.e. $c(v) = C, \forall v \in V_S \cup V_R$ (in the case of Netflix, $C = 10$ Gbps [net12]). We present now the algorithm that constructs delivery trees in a CDN with uniform equipment capacity C.

The algorithm is based on the following two principles : (i) use a single delivery tree for d_{ij} ; (ii) associate every reflector to at most one delivery tree. Then, the general idea is to deliver the representations in a greedy manner according to their potential "revenue" at the edge servers, i.e. we would like to deliver first the representations with the highest *utility score per rate unit* (`uspru`). For every edge server u and every representation d_{ij}, we compute the `uspru` of receiving d_{ij} at u as $\texttt{uspru}(u,d_{ij}) = \alpha_u^{ij}/\lambda_i$. The internals of representation d_{ij}, $i \in [k], j \in [l]$, delivery are as follows (illustrated in FIGURE 1). First d_{ij} is forwarded from some source node to a single reflector. Then, d_{ij} is disseminated through a tree

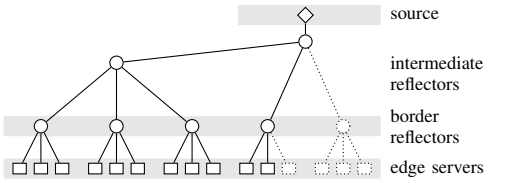

FIGURE 1: T_{ij} construction : $\delta_i = 3$, $m_{ij} = 4$, $g_{ij} = 11$, and 2 intermediate nodes.

structure of reflectors, until they reach the edge servers. We refer to the reflectors which are connected to the edge servers as the *border reflectors*. Due to the capacity bound of C, every reflector can only forward a representation to $\lfloor C/\lambda_i \rfloor$ other reflectors. We denote by δ_i this fan-out limitation. Also, let m_{ij} be the number of border reflectors in T_{ij}. Note that in order to serve g_{ij} edge servers with d_{ij} we need $m_{ij} = \lceil g_{ij}/\delta_i \rceil$ border reflectors.

Lemma 3.1. *In order to serve g_{ij} edge servers, $\lceil \frac{m_{ij}-1}{\delta_i - 1} \rceil$ intermediate reflectors are required in T_{ij}.*

Démonstration. For simplicity of notation we omit the subscript $(_{ij})$. First we create m border reflectors and connect them to the g edge servers, with at most δ connections at each border reflector. Next, we define U as the set of reflectors in T that currently do not have a parent in T ; in the beginning U holds all the border reflectors. We build T in an iterative bottom-up fashion. At every step we : (a) choose δ nodes from U, denote them as A (if $|U| < \delta$ then $A = U$) ; (b) create a new reflector v (c) connect v to every reflector $u \in A$; (d) update U by removing the subset A and adding v. This process continues until there is only one reflector in U, which is then connected to an arbitrary source node. It is easy to observe that the total number of intermediate reflectors used in the above construction is $\lceil \frac{m-1}{\delta - 1} \rceil$. □

Now we only need to decide on the number of edge servers g_{ij} that will receive d_{ij}, for every $i \in [k], j \in [l]$. We do it by iteratively going over the values of `uspru` in decreasing order. For every `uspru`(u, d_{ij}), if there are enough unused reflectors to support the possible increase in reflectors in T_{ij}, they are added to T_{ij}. The required increase in T_{ij} is computed according to Lemma 3.1 for $g_{ij} + 1$ edge servers. This continues until we reach the x^*-th `uspru` such that either all the requested representations are served to all the edge servers, or the representation in the considered `uspru` cannot be served due to lack of reflectors. Let $R(x)$ be the number of reflectors that have been allocated after the x highest `uspru`-s have been processed. We have :

$$R(x) = \sum_{i=1}^{k} \sum_{j=1}^{l} \left(\lceil g_{ij}(x)/\delta_i \rceil + \left\lceil \frac{\lceil g_{ij}(x)/\delta_i \rceil - 1}{\delta_i - 1} \right\rceil \right)$$

where $g_{ij}(x)$ is the number of edge servers with a processed `uspru` related to representation d_{ij}. Then,

$$x^* = \arg \max_{x \leq kl|V_E|} \{R(x) \leq |V_R|\}.$$

In the following, we provide formal theoretical analysis about the running time and the approximation ratio of our algorithm. For the running time, there are two major parts : (a) creating and sorting `uspru` ; (b) building delivery trees. The first part takes $O(k \cdot l \cdot |V_E| \log(k \cdot l \cdot |V_E|))$ time, whereas the second part is linear in $k \cdot l \cdot |V_E|$. Therefore the running time of our algorithm is $O(k \cdot l \cdot |V_E| \log(k \cdot l \cdot |V_E|))$. Then, we compare the utility score achieved by our algorithm S, and the best possible one (optimal) S^*. We denote by λ^* the bit-rate of the highest representation, that is $\lambda^* = \max_{i \in [k]} \lambda_i$.

Theorem 3.2. $S \geq \left(1 - \frac{3\lambda^*}{C} - \frac{2kl}{|V_R|}\right) S^*$.

Démonstration. We start by drawing an upper bound on S^*. We define $Q(x)$ the total capacity used by the representations that correspond to the first x `uspru`-s, and $\alpha(x)$ the sum of the utility scores that are fulfilled after the delivery of the first x `uspru` representations. The maximum available capacity at the reflectors is $|V_R|C$. Clearly, if there existed $x' \in X$ such that $Q(x') = |V_R|C$, then the highest achievable utility would have been at most $\alpha(x')$; this is because we deliver the representations in the decreasing order of `uspru`. However, x' might not exist ; let z^* be the last `uspru` entry such that $Q(z^*) \leq |V_R|C$. Let α_{z^*} be the utility score of the z^* `uspru`. We can obtain an upper bound, $S^* \leq \alpha(z^*) + (|V_R| \cdot C - Q(z^*))\, \alpha_{z^*}$.

Now, let us focus on evaluating the score S obtained by our algorithm. Our construction incurs some capacity wastage as a result of using intermediate reflectors. In every delivery tree T_{ij} there are m_{ij} border

reflectors, with at least $m_{ij}-1$ of them delivering d_{ij} to δ_i edge servers each. Let ϕ_{ij} be the capacity used in T_{ij} to deliver d_{ij}. We have $\phi_{ij} \geq (m_{ij}-1)\delta_i\lambda_i \geq (m_{ij}-1)(C-\lambda_i)$. Now, we have to consider the unused capacities. Let σ_{ij} be the capacity that is not used to deliver d_{ij} to the edge servers in T_{ij} ; this includes the overall capacity of intermediate nodes and the unused capacity by the border reflectors. According to Lemma 3.1 there are m_{ij} border reflectors and $\lceil \frac{m_{ij}-1}{\delta_i-1} \rceil$ reflectors in T_{ij}. We derive

$$\begin{aligned}\sigma_{ij} &= \left\lceil \frac{m_{ij}-1}{\delta_i-1} \right\rceil C + (m_{ij}C - \phi_{ij}) \leq \left(m_{ij} + \left\lceil \frac{m_{ij}-1}{\delta_i-1} \right\rceil \right) C - (m_{ij}-1)(C-\lambda_i) \\ &= \left\lceil \frac{m_{ij}-1}{\delta_i-1} \right\rceil C + m_{ij}\lambda_i + C - \lambda_i \leq 3m_{ij}\lambda_i + 2C.\end{aligned}$$

The last inequality is derived from the upper bound of intermediate reflectors bandwidth : $2m_{ij}\lambda_i + C$. Let Φ be the total capacity used by the algorithm to deliver the representations to the edge servers ; we have

$$\Phi = \sum_{i=1}^{k}\sum_{j=1}^{l} \phi_{ij} = |V_R|C - \sum_{i=1}^{k}\sum_{j=1}^{l} \sigma_{ij} \geq |V_R|C - 3|V_R| \max_{i\in[k]} \lambda_i - 2klC.$$

As the algorithm greedily delivers the `uspru` entries in decreasing order, the utility score is at least $\frac{\Phi}{|V_R|C}$ times the best possible one, which is the upper bound of S^*. We can therefore conclude,

$$S \geq \frac{\Phi}{|V_R|C} S^* \geq \frac{|V_R|C - 3|V_R|\lambda^* - 2klC}{|V_R|C} S^* = \left(1 - \frac{3\lambda^*}{C} - \frac{2kl}{|V_R|}\right) S^*.$$

□

4 Conclusions and future work

This paper depicts theoretical outcomes, but practical performances are excellent in realistic settings. For example upon the assumption of the highest representation bit-rate to be $\lambda^* = 7$ Mbps, and the capacity of reflectors to be $C = 10$ Gbps, with $k = 10$ (representations) and $l = 3$ (channels), the approximation ratio varies from 0.939 for a CDN with 500 reflectors, to 0.993 for larger CDNs with 5000 reflectors.

As future direction for this work, it would be interesting to propose solutions for the general problem. In addition, the overall performance can benefit from non-centralized computation of delivery trees. Finally, it is of great importance to explore the computation of the utility score, which is influenced by a large number of parameters, especially in the context of DASH where the demand from clients may change very quickly.

Références

[ASV11] Micah Adler, Ramesh K. Sitaraman, and Harish Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18) :4007–4020, 2011.

[Ing12] Mathew Ingram. You think the internet is big now ? akamai needs to grow 100-fold. Om Malik, Jun 2012. http ://is.gd/3vTZPC.

[net12] Netflix Open Connect Appliance Hardware, 2012. http ://is.gd/1aomd9.

[NSS10] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai network : a platform for high-performance internet applications. *Op. Sys. Rev.*, 44(3) :2–19, 2010.

[SLC+11] Sudipta Sengupta, Shao Liu, Minghua Chen, Mung Chiang, Jin Li, and Philip A. Chou. Peer-to-peer streaming capacity. *IEEE Trans. on Information Theory*, 57(8) :5072–5087, 2011.

[ZAB+12] Fen Zhou, Shakeel Ahmad, Eliya Buyukkaya, Gwendal Simon, and Raouf Hamzaoui. Minimizing Server Throughput for Low-Delay Live Streaming in Content Delivery Networks. In *ACM NOSSDAV*, 2012.

[ZLW11] Can Zhao, Xiaojun Lin, and Chuan Wu. The streaming capacity of sparsely-connected P2P systems with distributed control. In *INFOCOM*, pages 1449 – 1457, 2011.