HAL
open science

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

Jiayi Liu, Gwendal Simon

To cite this version:

Jiayi Liu, Gwendal Simon. Fast Near-Optimal Algorithm for Delivering Multiple Live Video Channels in CDNs. ICCCN 2013 : 22nd International Conference on Computer Communications and Networks, Jul 2013, Nassau, Bahamas. 10.1109/ICCCN.2013.6614138 . hal-00908757

HAL Id: hal-00908757

https://hal.science/hal-00908757v1

Submitted on 25 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Jiayi Liu Gwendal Simon
Télécom Bretagne, France, firstname.lastname@telecom-bretagne.eu

***Abstract*—Content Delivery Networks (CDNs) are confronted with a sharp increase in traffic related to live video (channel) streaming. Previous theoretical models that deal with streaming capacity problems do not capture the emerging reality faced by today's CDNs. In particular, a modern CDN has to deliver a large set of independent non-divisible data streams, which need to be either delivered in whole, or not delivered at all. This constraint is not addressed in previous works.**

In this paper we identify a new, discretized streaming model for live video delivery in CDNs. For this model we formulate a general optimization problem and show that it is NP-complete. Then we study a practical scenario that occurs in real CDNs. We present a fast, easy to implement, and near-optimal algorithm with performance approximation ratios that are negligible for large network. To our knowledge, these are the first results for the discretized streaming model, and have both practical and theoretical importance in a topic of growing criticality.

I. Introduction

In today's Internet, a small set of companies—referred to as Content Delivery Network (CDN) providers—handle the majority of video traffic on behalf of content producers. The techniques used by these companies to deliver terabytes of data per second are mainly exposed by reverse-engineering studies [1]. In comparison to the importance of video traffic today, the amount of research related to *live* video delivery in CDN infrastructures has stayed remarkably low.

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) [2], and the restriction on the upload capacity of the equipment [3, 4]. The goal of these previous works is to reduce the transmission cost of video delivery. However, modern CDNs rely on edge-servers that are located within the network of Internet Service Providers (ISPs), and on peering agreements with these ISPs [5]. As a matter of fact, the bandwidth cost to make the traffic transit across different networks has significantly decreased [6], to a point that it is no longer the main issue.

The major concern today is the growth in the volume of video traffic, and the capacity problem that this growth produces [7]. In addition, the development of *dynamic rate-adaptive streaming* techniques (e.g. MPEG DASH standard) accelerates the stress on the CDN. With such techniques, up to a dozen of *representations* of the same video stream are available at the server(s). Clients dynamically choose representation according to the accepted resolutions of their devices and their network capacity. This technique puts much stress on the infrastructure, since for a single channel the whole set of representations (with an aggregated bit-rate over 30 Mbps) should be delivered to the edge-servers.

The *streaming capacity* of networks has been addressed in a series of recent works [8, 9], which aim to determine the maximum bit-rate that can be delivered to all nodes in a network. Some algorithms, mostly based on network coding, obtain near-optimal performances in terms of bandwidth utilization [10]. Unfortunately, these solutions are unrealizable in a CDN due to two main reasons. First, they rely on heavy computations which are intractable in the CDN hardware (although the equipment have a very large bandwidth, their computing capabilities are quite small [5]). Second, the model used in these works is idealized since it assumes one infinitely divisible data stream, whereas the data that has to be delivered is a large set of distinct non-divisible streams (either representations, or *bundles* of representations). Each stream has to be either delivered in its entirety, or not delivered at all.

Discretized streaming is a more suitable model for multiple live video channels in modern CDNs. The main challenge is to determine a delivery scheme that maximizes the number of delivered streams in a 3-tier network that is constrained by the capacity of its inner equipment. We give a formal formulation of the general problem for this model and prove that it is NP-complete. This is the first contribution in this paper.

Then, we provide a solution that meets the demands of CDN providers. We consider a practical scenario, which corresponds to today's CDN implementation of live streams. We present an algorithm, which is fast, easy to implement, and near optimal. We provide formal theoretical approximation bounds, which are shown to be negligible for the regarded configuration. The algorithm represents the major contribution of this paper. To our knowledge, today's CDN providers apply ad-hoc delivery techniques. Our algorithm is thus the first scientific reference for optimal delivery in the discretized streaming model.

The remainder of the paper is as follows. In Section II, we introduce the discretized streaming model and define the problem. Then, in Section III we provide an ILP formulation of the problem and prove that it is NP-complete. In section IV, we overview the rationale behind the network configuration and describe our near-optimal algorithm for the scenario. Section V evaluates the performance of the proposed algorithm. Related work is presented in Section VI. Finally, Section VII discusses the theoretical results and concludes the paper.

II. System model and problem definition

A CDN is composed of a set of communication devices and a set of directed communication links. There are three types of communication devices, also referred to as nodes, in a CDN: a relatively small number of *sources* (origins), a medium size network of *reflectors*, and a large number of *edge servers*. The sources receive and transcode the raw video channels into a set of live *representations*; the reflectors deliver the representations to the CDN edges, and the edge servers offer the received representations to the clients inside their respective ISPs. In what follows we present our model of live streaming in a CDN, which is followed by a detailed optimization problem formulation.

A. Live video streaming in a CDN

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. 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 representations to the edge servers. They are formally defined as:

$$\begin{aligned} E_{SR} &= \{(u,v) : u \in V_S, v \in V_R\} \\ E_{RR} &= \{(u,v) : u, v \in V_R\} \\ E_{RE} &= \{(u,v) : u \in V_R, v \in V_E\}. \end{aligned}$$

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, $i \in [k]$, $j \in [l]$.

The delivery of a representation d_{ij}, $i \in [k]$, $j \in [l]$, from the source nodes to the edge servers is carried out through a set, $\mathcal{T}_{ij}$, of node-disjoint subtrees of G. Each tree in $\mathcal{T}_{ij}$, also referred to as the *delivery tree*, has one of the source nodes as its root and edge servers as its leafs. We denote by T_s^{ij} the delivery tree of d_{ij} rooted at $s \in V_S$. For convenience, let $V(T)$ and $E(T)$ denote the node and edge sets of tree T, respectively.

Note that every *forwarding node* v, either source or reflector, can participate in the delivery of multiple representations. However, for any representation d_{ij}, $i \in [k]$, $j \in [l]$, v can be a part of only a single delivery tree in $\mathcal{T}_{ij}$. In addition, every forwarding node $v \in V_S \cup V_R$ is also limited by the total outbound bit-rate (capacity) it can support, $c(v)$. Let $D(v)$ be the set of representations forwarded by v, $v \in V_S \cup V_R$. Then,

$$\sum_{i \in [k]} \lambda_i \cdot |\{j : d_{ij} \in D(v)\}| \leq c(v).$$

Like all previous works [3, 4, 11], we consider that the outbound capacity of equipment is the only constraint.

B. Problem definition

Ultimately we would like every edge server to receive all the representations it requires. This however might not be possible due to the outbound capacity constraints at the forwarding nodes, and thus the CDN may support the delivery of only a subset of representations for each edge server. In such case, the CDN provider leverages statistics to prioritize the delivery [2]. 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.

Our objective in this paper is to study the *Maximum Average Utility Score* (MAUS) problem, which essentially is the maximization of the *average* utility score function of the edge servers, as summarized below.

Problem II.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.*

III. Formulation and problem complexity

We now discuss the complexity of Problem II.1. We provide an ILP formulation and then show that it is NP-complete.

A. ILP formulation

We use the notation introduced in Section II and extend it by defining two new variables, y and h. Let $T_s^{ij} \in \mathcal{T}_{ij}$, $i \in [k]$, $j \in [l]$, be a delivery tree. Then, for every edge $(u,v) \in E$, $y_{u,v}^{ijs}$ is an indicator variable such that:

$$y_{u,v}^{ijs} = \begin{cases} 1 & \text{if } (u,v) \in E(T_s^{ij}), \\ 0 & \text{otherwise.} \end{cases}$$

For nodes $u, v \in V$ such that $(u,v) \notin E$ we define $y_{u,v}^{ijs} = 0$. For every node $v \in V$, h_v^{ijs} is an upper bound on the depth of v in T_s^{ij}, i.e.

$$h_{u,v}^{ijs} = \begin{cases} \geq \text{depth of } v \text{ in } T_s^{ij}, & \text{if } (u,v) \in E(T_s^{ij}), \\ = \infty, & \text{otherwise.} \end{cases}$$

To ease the notation, let us define $I_v^{ijs}(U)$ to be the sum of y variables that correspond to incoming edges into $v \in V$ from the nodes in $U \subseteq V$, i.e.

$$I_v^{ijs}(U) = \sum_{u \in U} y_{u,v}^{ijs}.$$

Similarly, let $O_v^{ijs}(U)$ be the sum of y variables that correspond to outgoing edges from v to nodes in U, i.e.

$$O_v^{ijs}(U) = \sum_{u \in U} y_{v,u}^{ijs}.$$

In the ILP formulation, we omit the use of set membership indication $\in$ for the main notations. Whenever we write $\forall i$, $\forall j$, $\forall s$, $\forall r$, $\forall u$, and $\forall v$, we imply $\forall i \in [k]$, $\forall j \in [j]$, $\forall s \in V_S$, $\forall r \in V_R$, $\forall u \in V_E$, and $\forall v \in V$, respectively.[1]

ILP formulation: MAUS

$$\textbf{max.} \quad \sum_{u \in V_E} \sum_{i=1}^{k} \sum_{j=1}^{l} \alpha_u^{ij} x_u^{ij} \tag{1}$$

$$\textbf{s.t.} \quad x_u^{ij} \leq \sum_{s \in V_S} I_u^{ijs}(V_R) \qquad \forall i,j,u \tag{2}$$

$$I_r^{ijs}(V_S \cup V_R) \leq 1 \qquad \forall i,j,s,r \tag{3}$$

$$I_u^{ijs}(V_R) \leq 1 \qquad \forall i,j,s,u \tag{4}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{l} O_s^{ijs}(V_R)\lambda_i \leq c(s) \qquad \forall s \tag{5}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{l} \sum_{s \in V_S} O_r^{ijs}(V_R \cup V_E)\lambda_i \leq c(r) \qquad \forall r \tag{6}$$

$$h_s^{ijs} = 0 \qquad \forall i,j,s \tag{7}$$

$$h_r^{ijs} + 1 - h_v^{ijs} \leq |V|(1 - y_{r,v}^{ijs}) \qquad \forall i,j,r,v \tag{8}$$

$$O_r^{ijs}(V_R \cup V_E) \leq |V|(I_r^{ijs}(\{s\} \cup V_R)) \quad \forall i,j,s,r \tag{9}$$

$$I_r^{ijs}(\{s\} \cup V_R) \leq O_r^{ijs}(V_R \cup V_E) \qquad \forall i,j,s,r \tag{10}$$

The constraints in (2) ensure that the indicator variables x have non-zero values only if there are incoming edges in the respective trees. Constraints in (3)-(4) guarantee that every node has only one parent in every delivery tree. Cycles are avoided in (7)-(8). The capacity restrictions are enforced in (5)-(6). Finally, in (9)-(10) we require that reflector nodes have outgoing edges in delivery trees iff there is an incoming edge.

B. NP-completeness

Let DMAUS be the decision version of the MAUS problem.

Problem III.1 (DMAUS). *Given topology and capacity constraints of a CDN, and a real number B, do there exist delivery tree sets, $\{\mathcal{T}_{ij}\}_{i \in [k], j \in [l]}$, such that $\sum_{u \in V_E} \alpha_u(X_u) \geq B$?*

Clearly DMAUS is in NP. We now show that DMAUS is NP-hard by a reduction from 3-SAT. Recall that an instance of the 3-SAT problem consists of n variables, $z_1, \ldots, z_n$, and m clauses, $C_1, \ldots, C_m$, where each clause $C_j = (y_j^1 \vee y_j^2 \vee y_j^3)$, $j \in [m]$, has exactly three literals.

Given an instance of the 3-SAT problem we construct an instance of the DMAUS problem. Let $G_{3SAT} = (V, E)$ be the topology of a CDN. We define V to be (i) a single source node, $V_S = \{s\}$, (ii) $3n$ reflectors that are partitioned into two sets, $V_R = Z \cup A$, such that $|Z| = n$ and $|A| = 2n$, (iii) and m edge servers $V_E = \{u_1, \ldots, u_m\}$. The node set $Z = \{v_1, \ldots, v_n\}$ represents the variables of the 3-SAT instance, the nodes in $A = \{v_1^t, v_1^f, \ldots, v_n^t, v_n^f\}$ represent the two possible values of these variables, and the edge servers represent the m clauses. Overall $|V| = 1 + 3n + m$.

The edge set E is composed of n links between s and the nodes in Z, $2n$ links that connect Z to A, and $3m$ links between A and the edge server nodes. More specifically,

$$\begin{aligned} E = &\{(s, v) : v \in Z\} \\ &\cup \{(v_i, v_i^t), (v_i, v_i^f) : i \in [n]\} \\ &\cup \{(v_i^t, u_j) : i \in [n], j \in [m], z_i \text{ is a literal in } C_j\} \\ &\cup \{(v_i^f, u_j) : i \in [n], j \in [m], \bar{z}_i \text{ is a literal in } C_j\} \end{aligned}$$

For example, see Fig. 1. The capacities of the nodes are defined as follows: $c(s) = n$ and $\forall i \in [n]$, $c(v_i) = 1$ and $c(v_i^t) = c(v_i^f) = m$. We set the number of channels and representations to 1, and the utility score of receiving the single available representation at every edge server $u \in V_E$ is $\alpha_u^{11s} = 1$. Finally, the value B is chosen to be m.

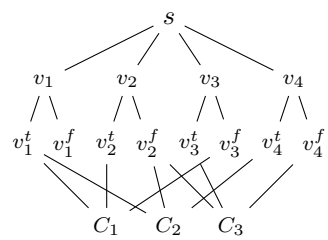


Fig. 1: Graph associated with 3-SAT $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4})$

We now show that there is a solution to the 3-SAT problem iff there is a solution to the DMAUS problem. Due to space constraints we provide only a brief outline of the proof.

Let ϕ be a satisfying assignment for the 3-SAT problem instance. We incrementally construct the delivery tree $T_s^{11} = (V_s^{11}, E_s^{11})$. At first $V_s^{11} = \{s\}$ and $E_s^{11} = \emptyset$. Then, if there exists an edge server $u_j \notin V_s^{11}$, we pick one of the literals in C_j that have a true assignment in ϕ (since ϕ is a satisfying assignment, there must be at least one such literal). W.l.o.g. let z_i be a literal in C_j and $\phi(z_i) = true$. We add to the tree the nodes v_i, v_i^t, u_j and the edges (s, v_i), (v_i, v_i^t), and (v_i^t, u_j). Note that some of the nodes or the edges might already exist in the tree, so we just add the missing ones. These steps never violate the capacity constraints, as for any $v_i \in Z$, at most one outgoing edge, either (v_i, v_i^t) or (v_i, v_i^f), can be added to E_s^{11}, which depends on the assignment ϕ (the first in case $\phi(z_i) = true$, and the latter otherwise). It is easy to conclude that T_s^{11} is a delivery tree rooted at s and has all the nodes V_E as its leafs, and thus the utility score function has a total score of m. The feasibility of each step follows directly from the definition of the CDN topology, G_{3SAT}.

In the opposite direction, let T_s^{11} be a solution to the DMAUS problem (as there is only one representation, one channel and one source node, the solution is a single delivery tree). We construct ϕ by iterating over the nodes in V_E. For every $u_j \in V_E$, if v_i^t is the parent of u_j in T_s^{11} we define $\phi(z_i) = true$, and $\phi(z_i) = false$ otherwise. Note that due to capacity constraints, it is impossible that both v_i^t and v_i^f are in

[1] We use i, j, s, r, u, and v to refer to representations, channels, sources, reflectors, edge servers, and general nodes, respectively.

T_s^{11}, and thus the assignment is feasible, i.e. the same variable will never be assigned both $true$ and $false$. After the iteration ends, if there are any undefined variables, we set their values to $true$. What remains to be shown is that ϕ is a satisfying assignment. For every clause C_j, $j \in [m]$, there must exist a literal which corresponds to the father of u_j in T_s^{11} (due to the construction of G_{3SAT}) and has a $true$ assignment in ϕ (due to the iterative definition of ϕ), and thus C_j is true.

IV. Homogeneous bundle delivery

As the above NP-completeness claim implies, it is currently impossible to implement an optimal solution for the general case. Thus, we focus on a practical scenario, which, as far as we understood from our discussions with CDN stakeholders, corresponds to today's CDN implementation. For this practical scenario, we propose a near optimal greedy algorithm which produces delivery trees for every channel.

A. Practical bundle delivery in CDN

In practical CDNs, one can assume that every reflector is connected to any other reflector by a direct link, i.e. for any $u, v \in V_R$ and $u \neq v$, it holds $(u, v) \in E_{RR}$. This can be justified by the fact that the links between reflectors are essentially international connections in the public Internet backbone, where any equipment is virtually connected to any other equipment. In fact, the specifications of the CDN Federation [12, 13] impose full connectivity between the hosts of every member CDN. In what follows we describe a fundamental and popular CDN scenario, named *Homogeneous Bundle Delivery*.

For services which are based on DASH, today's CDNs do not deliver each representation individually. Instead, they gather all representations of a given channel into one *bundle*, and deliver the whole bundle from the source(s) to the edge server(s). Due to the fact that the majority of transcoders are the same, all bundles have roughly the same size, which simplifies the delivery management.

An example of this scenario is as follows: the client of a large-scale CDN is a prominent over-the-top (OTT) service provider, which diffuses a TV package to a large audience. Every channel is bundled, with a total rate of λ. The number of channels l ranges typically from 20 to 150. All the edge servers are expected to receive all the bundles in the same way, i.e. $\alpha_u^j = 1$ for every $u \in V_E$ and $j \in [l]$ (note the slight change of notation due to the bundling of representations). For a fixed rate data stream, the capacity constraint of every node is essentially an upper bound on the number of simultaneous bundles that the node can support. For simplicity let $b_v = \lfloor c(v)/\lambda \rfloor$ be the number of bundles that can be supported by any $v \in V_S \cup V_R$. As previously we use the indicator variables x_u^j, $u \in V_E$, $j \in [l]$, which have the value of 1 if the edge server u receives the bundle of the j-th channel. Our objective for this scenario can be summarized as follows:

$$\max \sum_{j \in [l]} \sum_{u \in V_E} x_u^j.$$

B. The algorithm

For simplicity of exposition, and following our assumption of full connectivity between the source and reflector nodes (E_{SR}), we can assume there is a single super-source s^* with $b_{s^*} = \sum_{s \in V_S} b_s$. Note that any solution for the case of having a single super-source can be easily converted into a solution with multiple sources by distributing the load among the sources according to their upload capacities.

The algorithm Bundle-Delivery (see below) is composed of two phases. First (step (2)), we iteratively construct one delivery tree T_j for every channel j, $j \in [l]$. Second (step (3a)), we provide a local improvement for potentially unused capacity. The first phase is further divided into two parts: first we decide which reflectors $V_j \subseteq V_R$ will be part of T_j (the loop in step (2c)); then, we generate the tree T_j in step (2f).

The main idea of the algorithm is to have as few reflectors in the delivery trees as possible. By doing so we aim to reduce the capacity "wasted" on inter-reflector communication. We also avoid using nodes with residual capacity of 1 in the first phase since they can forward a bundle to only one node. Instead, we use 2-hop connections in the second phase to utilize their capacity.

There are two main sets of variables in the algorithm. For every node $v \in V_R$ we use b_v^j, $0 \leq j \leq l$, to denote the residual forwarding capacity of node v after the construction of trees $T_1, \ldots, T_j$, i.e. the capacity which remains at v to forward bundles for channels $j+1, \ldots, l$ after it has already forwarded the bundles for channels $1, \ldots, j$. We also define f_v^j, for every $v \in V_R$, $j \in [l]$, to be the number of bundles forwarded by v in T_j, i.e. the number of children v has in T_j. After V_j is determined, these variables are updated according to the forwarding capacity used by each node in V_j. The formal definition of Bundle-Delivery follows.

We now explain steps (2f) and (3a) in detail.

Algorithm: Bundle-Delivery

1) Initialize $\forall v \in V_R : b_v^0 \leftarrow b_v$ and $j \leftarrow 1$.
2) While $\exists v \in V_R : b_v^{j-1} \geq 2$, $b_{s^*} \geq j$, and $j \leq l$:
   a) Initialize $V_j \leftarrow \emptyset$ and $U_j = \{u : b_u^{j-1} \geq 2\}$.
   b) Initialize $\forall v \in V_R : f_v^j \leftarrow 0$.
   c) While $\sum_{v \in V_R} f_v^j < |V_j| + |V_E| - 1$ and $U_j \neq \emptyset$:
      i) Extract from U_j a node u^* with maximum forwarding capacity, i.e. $b_{u^*}^{j-1} = \max_{u \in U} b_u^{j-1}$.
      ii) $f_{u^*}^j \leftarrow \min\{b_{u^*}^{j-1}, |V_E| + |V_j| - \sum_{v \in V_j} f_v^j\}$.
      iii) Add u^* to V_j.
   d) Update $\forall v \in V_R : b_v^j = b_v^{j-1} - f_v^j$.
   e) Update $j \leftarrow j + 1$.
   f) **Generate a tree** T_j based on the nodes V_j.
3) If not all edge servers receive all channel bundles and $b_{s^*} > l$ then:
   a) **Use two-hop connections** to deliver additional channel bundles.

Generating a tree. For every channel j, T_j is a tree which

is rooted at s^*, has V_j as its intermediate nodes, and some or all of V_E as its leafs. The out-degree of every node $v \in V_j$ in T_j is exactly f_v^j. The topology of the tree can be arbitrary with a single constraint, that the super-source has exactly one child in T_j. Although the topology of T_j has no effect on the number of leafs in T_j, which is $(\sum_{v \in V_j} f_v^j - |V_j| + 1)$ as we show later, we would ultimately like the tree to have the minimum possible height to minimize the number of hops from the super-source node to the leafs. For that purpose we construct T_j in the following way. First connect s^* to the node v with maximum value of f_v^j in V_j. Then, connect v to f_v^j nodes with the next highest values of f^j in V_j, and repeat this process for every node in $V_j \setminus \{v\}$ according to decreasing value of f^j until all the nodes in V_j have a parent in T_j. In the end of this process, some nodes will have an out-degree less than the corresponding value of f^j. Connect these nodes to a subset of edge servers, yet to be included in T_j, such that the degree of those nodes will match their f^j values (Lemma IV.1 below shows that it is always possible).

Fig. 2 shows an example of tree generation. The node-set V_j is composed of two nodes: u and v with $f_u^j = 3$, and $f_v^j = 2$ (Fig. 2(1)). As u has a higher forwarding capacity than v, it is connected to s^* and v is set as the child of u (Fig. 2(2)). At this point the out-degrees of u and v are 1 and 0, respectively, which are less than their forwarding capacities in T_j. Thus, both u and v are connected to 2 edge servers each (Fig. 2(3)).

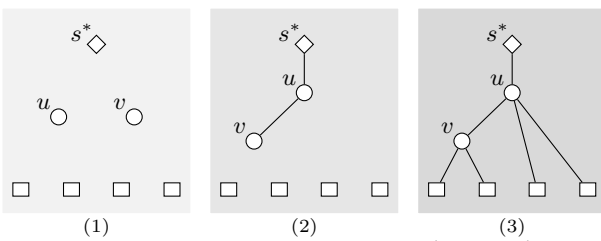


Fig. 2: T_j generation: $V_j = \{u, v\}$, $f_u^j = 3$, $f_v^j = 2$.

Using two-hop connections: After step (2), the source node might have some unused forwarding capacity. As we show later, if not all edge servers receive every channel bundle, then every reflector $v \in V_R$ has a forwarding capacity of $b_v^{j^*} \leq 1$, where j^* is the last channel for which a tree T_j is constructed (the last execution of the step (2)). We use reflectors with non-zero forwarding capacity to deliver additional channel bundles to edge servers that are yet to receive them in a two-hop fashion $s^* \rightarrow v \rightarrow u$, where $v \in V_R$ with $b_v^{j^*} = 1$, and $u \in V_E$ such that u is not a leaf in T_i.

Let j^* be the last channel for which a delivery tree T_{j^*} was constructed. It is easy to observe that the trees generated in every step (2f) are feasible delivery trees rooted at s^*, where T_j delivers the j-th channel bundle, $j \in [j^*]$. The forwarding capacity upper bound is enforced for the super-source in step (2) for the reflectors in steps (2(c)ii,2d). Moreover, as we show in Lemma IV.1 below, all the leafs in T_j are edge server nodes.

Lemma IV.1. *In every constructed tree T_j there are exactly $\sum_{v \in V_j} f_v^j - (|V_j| - 1)$ leafs and $\sum_{v \in V_j} f_v^j - (|V_j| - 1) \leq |V_E|$.*

Proof: Note that for any T_j, $j \in [l]$, the root s^* has exactly one child. Therefore, $|V_j| - 1$ nodes have a parent node which is a reflector. According to step (2f), every reflector $v \in V_j$ has an out-degree of f_v^j in T_j. Thus, we can conclude that in T_j a total of $|V_j| - 1$ forwarding capacity is used to deliver the channel bundle between reflector nodes, which results in T_j having $\sum_{v \in V_j} f_v^j - (|V_j| - 1)$ leaf nodes. Steps (2c,2(c)ii) enforce the inequality $\sum_{v \in V_j} f_v^j - (|V_j| - 1) \leq |V_E|$, and therefore we can conclude that step (2f) is feasible, i.e. it is always possible to connect a reflector v to some edge server, yet to be in T_j, to fill the out-degree of v in T_j. ■

C. Performance analysis of BUNDLE-DELIVERY

Running time. The algorithm BUNDLE-DELIVERY has three main steps. Step (1) is simple initialization and takes $O(|V_R|)$ time to execute. In step (2) we need to maintain the information about the residual forwarding capacity at every reflector. This can be easily implemented by using an ordered list. At first the reflectors are sorted in decreasing order according to their initial forwarding capacity (b_v^0, $v \in V_R$). Then, during the execution of the inner loop in step (2(c)ii), for every $j \in [l]$ at most one node in V_j will not use all of its forwarding capacity in T_j. Removing all the nodes except for, possibly, the last one, and moving the last one in the ordered list according to its updated residual capacity, takes $O(|V_R|)$ time. Clearly the use of the ordered list allows an easy implementation of the collection of nodes U_j, as we are only interested in the information about the residual capacities in decreasing order. Tree generation itself takes linear time in $|V_j| + |V_E|$. Thus, the total running time of the second step is $O(|V_R| \log |V_R| + l \cdot V_R + |V_E|)$. Finally, the third step takes $O(b_{s^*} + |V_R| + |V_E|)$ time. To conclude, the running time of BUNDLE-DELIVERY is $O(|V_R| \log |V_R| + l \cdot V_R + |V_E| + b_{s^*})$.

Approximation ratio. In our analysis we ignore step (3a) of the algorithm as it has no direct effect on the performance bounds, but rather serves as a local improvement, which may or may not occur. Let S and S^* be the values of the solution obtained by BUNDLE-DELIVERY and the optimal one, respectively. The next lemma shows that either the unused capacity of every reflector is at most 1 or $S = S^*$.

Lemma IV.2. *At the end of the execution of* BUNDLE-DELIVERY *it holds that if $\exists u \in V_R : b_u^{j^*} > 1$ then $S = S^*$.*

Proof: Suppose that after the construction of T_{j^*} there exists a node $u \in V_R$ such that $b_u^{j^*} > 1$. Clearly, $u \in U_j$ for every $j \in [j^*]$ as the residual forwarding capacity $b_u^{(\cdot)}$ can only decrease in subsequent executions of step (2). There are two possible cases during the construction of T_j:

Case 1: Node u was chosen in step (2(c)i). Then since $b_u^{j^*} > 1$ we can conclude that $f_u^j = |V_E| + |V_j| - \sum_{v \in V_j \setminus \{u\}}$ and u is the last node to be added to V_j.

Case 2: Node u was not chosen in step (2(c)i). Then, the loop (2c) ended due to equality $\sum_{v \in V_j} f_v^j = |V_E| + |V_j| - 1$.

Thus, for every $j \in [j^*]$ the equality $\sum_{v \in V_j} f_v^j - (|V_j| - 1) = |V_E|$ holds and due to Lemma IV.1 the number of leafs in T_j is $|V_E|$. Taking a closer look at the conditions in the main loop (2) we can see that $j^* = \min\{l, s^*\}$ (as for any

$j \in \{0, 1, \ldots, j^* + 1\}$, $b_u^{j-1} \geq 2$), and as a result $S = |V_E| \cdot \min\{l, s^*\}$. On the other hand, we have $S \leq S^* \leq |V_E| \cdot \min\{l, s^*\}$. Therefore, $S = S^*$. ■

We are now ready to prove the main theorem of this section. We make a reasonable assumption that it is possible to deliver at least one channel to all the edge servers, i.e. $S^* \geq |V_E|$.

Theorem IV.3. *Under the assumption that $S^* \geq |V_E|$ it holds that $S/S^* \geq 1 - (b_{s^*}/|V_E|)$.*

Proof: We start by drawing an upper bound on the optimal solution. The number of edge servers that can potentially be reached by all the reflectors is at most $\sum_{v \in V_R} b_v$. However, some of the capacity needs to be used to maintain the delivery trees (source-reflector, and reflector-reflector connections). In the best case scenario (in terms of "wasted" capacity), every node is used in only one tree, and b_{s^*} reflectors have s^* as their parent in one of the delivery trees. Thus, $S^* \leq \sum_{v \in V_R} b_v - |V_R| + b_{s^*}$.

Next we analyze the solution produced by BUNDLE-DELIVERY. If there exists a node $v \in V_R$ such that $b_v^{j^*} > 1$ (recall that T_{j^*} is the last tree to be constructed), then according to Lemma IV.2, $S = S^*$. Otherwise, for every $v \in V_R$, $b_v^{j^*} \leq 1$. Let $x = |\{v : b_v > 1, b_v^{j^*} = 1\}|$ be the number of nodes that had their residual bundle capacity reduced to 1 during the execution of the algorithm, and $y = |\{u : b_u = 1\}|$ be the number of nodes with bundle delivery capacity of 1 prior to the execution of the algorithm. Hence the total forwarding capacity used in all the delivery trees (before step (3a), $\sum_{j \in [j^*]} \sum_{v \in V_j} f_v^j = \sum_{v \in V_R} b_v - (x + y)$.

Note that when a node u^* is selected to be added to V_j, $j \in [j^*]$, in step (2(c)i) it cannot be used again unless it was the last node to be added to T_j (due to the computation of the forwarding capacity $f_{u^*}^j$). Thus, the total number of reflectors in all the delivery trees is the number of potentially participating nodes ($v \in V_R$, with $b_v > 1$) plus at most $j^* - 1$ times a node might appear in two and more trees (due to the partial assignment of f). Formally, $\sum_{j \in [j^*]} |V_j| \leq |V_R| - y + j^* - 1$.

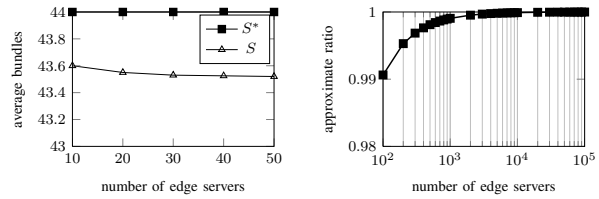
Based on Lemma IV.1 and the above we can now derive a lower bound for our solution. As discussed in the beginning of this section we ignore step (3a) in our evaluation (it can only improve the lower bound of S^*).

$$\begin{aligned} S &= \sum_{j \in [j^*]} \left(\sum_{v \in V_j} f_v^j - (|V_j| - 1) \right) \\ &= \sum_{j \in [j^*]} \sum_{v \in V_j} f_v^j - \sum_{j \in [j^*]} (|V_j| - 1) \\ &\geq \sum_{v \in V_R} b_v - (x + y) - (|V_R| - y + j^* - 1) + j^* \\ &= \sum_{v \in V_R} b_v - |V_R| - x + 1 \end{aligned}$$

As we derived in Lemma IV.2, when a partial forwarding capacity $f_v^j < b_v^{j-1}$ is assigned to some $v \in V_j$, $j \in [j^*]$, the delivery tree T_j has $|V_E|$ leafs. As we already stated in this proof, at most one node can be assigned a partial forwarding capacity in each tree, and thus there is a partial assignment in at least x delivery trees. As a result, $S \geq x|V_E|$. On the other hand, $S \leq S^* \leq l|V_E|$. Summarizing all of the above and under the assumption that $S^* \geq |V_E|$ we obtain,

$$\begin{aligned} S/S^* &\geq \frac{S^* - x - b_{s^*} + 1}{S^*} = 1 - \frac{x + b_{s^*} - 1}{S^*} \\ &= 1 - \frac{1}{|V_E|} - \frac{b_{s^*} - 1}{S^*} \geq 1 - (b_{s^*}/|V_E|). \end{aligned}$$

■



(a) Small instances (b) Large instances

Fig. 3: Evaluation

V. EVALUATION

We now evaluate the approximation ratio, S/S^* of the BUNDLE-DELIVERY algorithm proposed in Section IV. For small instances, S^* is obtained by the implementation of the ILP model in IBM ILOG CPLEX software. Due to the complexity of the model, for large instances, we computed S^* according to the upper bound given in the proof of Theorem IV.3. We simulated a CDN network with 1 source, x edge servers (x between 10 and 100,000), and the number of reflectors can supply 90% of the required bandwidth to deliver 50 channels. Each channel contains 8 representations. The bit-rate of the representations follows recommendations from Apple HTTP Live Streaming [14]: $\{150, 240, 440, 640, 1240, 1840, 2540, 4540\}$ kbps. Source and reflector capacity is set to 1 Gbps.

The results are shown in Figure 3. For small networks (Figure 3(a)), edge servers receive 44 channels on average in the optimal solution. The result obtained by the BUNDLE-DELIVERY algorithm is slightly below. For large networks (Figure 3(b)), the BUNDLE-DELIVERY algorithm achieves an approximation ratio of $1 - 10^{-3}$. For networks with 1,000 edge servers, the ratio S/S^* is at least 0.999056, and starting from $x = 10{,}000$ it is above 0.999906.

VI. RELATED WORK

In what follows we survey some of the relevant literature in related areas.

Streaming in CDN networks. A surprisingly low amount of work are related to *live* streaming in CDN networks. The earliest work [15, 16] did not deal with multiple streams, which is the main challenge in our work. The most recent work [4, 11] consider the lack of resources and multiple streams, however their objective is to reduce the bandwidth cost subject to the resource constraints. This objective is outdated since CDNs and ISPs develop peering agreements, which reduce the

importance of the bandwidth cost. Finally, the relation between edge servers and end users in the context of DASH is studied in [17, 18]. These work complement our work, which focuses on the CDN infrastructure.

Multi-tree Packing Delivery. Bounded tree packing problems have been studied in the context of peer-assisted systems [19, 20]. The problem is to minimize the amount of additional resources to serve all peers in a peer-to-peer system. This problem is different in CDNs, which are self-sustained networks; missing resources cannot be compensated, but rather need to be used in the best possible way. Numerous work have studied multi-tree packing for peer-to-peer application-layer multicast protocols (see [21] for a survey). The goal here is to span *all nodes* under application-related optimization objective (*e.g.*, to minimize tree height, or to reduce controlling overhead).

Streaming capacity in node-capacitated networks. The problem of maximizing the rate of the data stream, subject to the upload capacities of the nodes has been addressed under different models [8, 9]. These work are motivated by the development of peer-to-peer networks. Again, the network and the delivery objectives differ. More important, these work do not address the problem of discretized streaming, and they do not deal with the delivery of particular channels. The maximization of channel utility scores is a major difference between these work and ours, which prevents the use of network coding techniques.

Bounded-degree Trees. The minimum Bounded Degree Spanning Tree (BDST) problem aims to determine a minimum-cost spanning tree while no node should have more than m children (see [22]), which is NP-complete for any $m \geq 2$. Related variations of this problem feature non-uniform degree bounds [23]. Our problem formulation differs since, first, consider an unweighted graph, and second, these work aim at spanning all nodes in the network while optimizing an objective function, while we aim at maximizing the number of spanned nodes under a node degree constraint. The only related work in this aspect is [24], which study the minimum spanning tree with at least k nodes in a weighted graph, but this work do not target the maximal k. Furthermore, these work do not deal with packing several trees and the resource allocation problem that such packing introduces.

VII. Conclusions and future work

In this paper we introduced the discretized streaming model, which represents multiple rate-adaptive video channels delivery in today's CDNs. We first formulated a general optimization problem for prioritized live video delivery which we showed to be NP-complete. Then, we focused on a realistic scenario. The CDN provider is in charge of delivering the representations of each channel as one bundle. We developed a fast and simple algorithm which guarantees a solution which is at least $1 - (b_{s^*}/|V_E|)$ times the optimal solution. To the best of our knowledge, this is the first result for the discretized streaming model

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.

Acknowledgment

We thank Dr. Hanan Shpungin for his contribution and French Ministry of Industry for the funding of *Zewall* project.

References

[1] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting Video Server Selection Strategies in the YouTube CDN," in *IEEE ICDCS*, 2011.

[2] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: a platform for high-performance internet applications," *Op. Sys. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[3] F. Zhou, S. Ahmad, E. Buyukkaya, G. Simon, and R. Hamzaoui, "Minimizing Server Throughput for Low-Delay Live Streaming in Content Delivery Networks," in *ACM NOSSDAV*, 2012.

[4] M. Adler, R. K. Sitaraman, and H. Venkataramani, "Algorithms for optimizing the bandwidth cost of content delivery," *Computer Networks*, vol. 55, no. 18, pp. 4007–4020, 2011.

[5] "Netflix Open Connect Peering Guidelines," 2012, http://goo.gl/GQ6NU.

[6] B. Krogfoss, "Analysis: Content peering and the internet economy," Alcatel Lucent, Tech. Rep., April 2011.

[7] M. Ingram, "You think the internet is big now? akamai needs to grow 100-fold," Om Malik, Jun. 2012, http://goo.gl/vVUap.

[8] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Peer-to-peer streaming capacity," *IEEE Trans. on Information Theory*, vol. 57, no. 8, pp. 5072–5087, 2011.

[9] C. Zhao, X. Lin, and C. Wu, "The streaming capacity of sparsely-connected P2P systems with distributed control," in *INFOCOM*, 2011.

[10] D. Niu and B. Li, "Asymptotic optimality of randomized peer-to-peer broadcast with network coding," in *IEEE INFOCOM*, 2011.

[11] K. Andreev, B. Maggs, A. Meyerson, J. Saks, and R. Sitaraman, "Algorithms for constructing overlay networks for live streaming," *CoRR*, vol. 1109.4114, 2011.

[12] G. Bertrand, E. Stephan, T. Burbridge, P. Eardley, K. Ma, and G. Watson, "Use Cases for Content Delivery Network Interconnection," draft at IETF CDN Interconnection, 2012.

[13] F. Le Faucheur, "CDN Federations: Lessons from the CDN Federation Pilot Phase 2," in *CDN Summit*, 2012.

[14] Apple, "Using http live streaming," http://goo.gl/fJIwC.

[15] K. Andreev, B. M. Maggs, A. Meyerson, and R. K. Sitaraman, "Designing overlay multicast networks for streaming," in *ACM SPAA*, 2003.

[16] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. J. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Trans. on Multimedia*, vol. 6, no. 2, pp. 356–365, 2004.

[17] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over http in content distribution network," *Sig. Proc.: Image Comm.*, vol. 27, no. 4, pp. 288–311, 2012.

[18] L. D. Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *ACM MMSys*, 2011.

[19] M. R. Rahimi, A. Bais, and N. Sarshar, "On fair and optimal multi-source ip-multicast," *Computer Networks*, 2012.

[20] R. Sweha, V. Ishakian, and A. Bestavros, "AngelCast: Cloud-based Peer-Assisted Live Streaming Using Optimized Multi-Tree Construction," in *ACM MMSys*, 2012.

[21] M. Hosseini, , D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Com. Surveys & Tut.*, vol. 9, no. 3, pp. 58–74, 2007.

[22] M. X. Goemans, "Minimum bounded degree spanning trees," in *IEEE FOCS*, 2006.

[23] J. Könemann and R. Ravi, "Primal-dual meets local search: approximating MSTs with nonuniform degree bounds," *SIAM J. Comput.*, vol. 34, no. 3, pp. 763–773, 2005.

[24] C. Blum and M. J. Blesa, "New metaheuristic approaches for the edge-weighted -cardinality tree problem," *Computers & OR*, vol. 32, pp. 1355–1377, 2005.