Minimizing Server Throughput for Low-Delay Live Streaming in Content Delivery Networks

Fen Zhou
Telecom Bretagne
Brest, France
fen.zhou@telecom-bretagne.eu

Shakeel Ahmad
De Montfort University
Leicester, United Kingdom
sahmad@dmu.ac.uk

Eliya Buyukkaya
Telecom Bretagne
Brest, France
eliya.buyukka@telecom-bretagne.eu

Raouf Hamzaoui
De Montfort University
Leicester, United Kingdom
rhamzaoui@dmu.ac.uk

Gwendal Simon
Telecom Bretagne
Brest, France
gwendal.simon@telecom-bretagne.eu

ABSTRACT

Large-scale live streaming systems can experience bottlenecks within the infrastructure of the underlying Content Delivery Network. In particular, the "equipment bottleneck" occurs when the fan-out of a machine does not enable the concurrent transmission of a stream to multiple other equipments. In this paper, we aim to deliver a live stream to a set of destination nodes with minimum throughput at the source and limited increase of the streaming delay. We leverage on rateless codes and cooperation among destination nodes. With rateless codes, a node is able to decode a video block of k information symbols after receiving slightly more than k encoded symbols. To deliver the encoded symbols, we use multiple trees where inner nodes forward all received symbols. Our goal is to build a diffusion forest that minimizes the transmission rate at the source while guaranteeing on-time delivery and reliability at the nodes. When the network is assumed to be lossless and the constraint on delivery delay is relaxed, we give an algorithm that computes a diffusion forest resulting in the minimum source transmission rate. We also propose an effective heuristic algorithm for the general case where packet loss occurs and the delivery delay is bounded. Simulation results for realistic settings show that with our solution the source requires only slightly more than the video bit rate to reliably feed all nodes.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems, trees*

General Terms

Algorithms, Design, Performance

Keywords

Rateless Codes, Live Streaming, Delivery Trees, CDN

1. INTRODUCTION

As illustrated by the recent decision from Korea Telecom to block Samsung's Smart TVs from accessing the Internet [7], *over-the-top live streaming* represents a major challenge for Internet Service Providers (ISPs) and Content Delivery Networks (CDNs). The bottleneck of large-scale live streaming delivery platforms is no longer in the *last-mile* since rate-adaptive streaming enables a match between video bit-rate and available bandwidth [8]. It is not in the *peering link* either because ISPs and CDNs develop more friendly relationships, including co-location of edge servers. Today, the bottleneck is likely located *within the CDN infrastructure*, which has to cope with the tremendous growth of video traffic and the multiplication of video encodings per TV channel. The (relative) failure of the latest SuperBowl streaming confirms this trend [13].

For live streaming, the infrastructure of a CDN comprises three distinct elements. The role of the *origin server* is to forward the original stream to *edge servers*. However the origin server has a limited fan-out, *i.e.,* the number of streams it can concurrently emit is limited. The CDN thus uses some *intermediate nodes*, which are called reflectors [3] or shield caches [5]. While these intermediate nodes have to serve a subset of the thousands of edge-servers, they have a limited upload capacity. The bottleneck in the CDN infrastructure comes either from the origin server that cannot serve the intermediates nodes, or from the intermediate node that cannot serve the edge servers.

In this paper, we introduce the generic problem of one source (either origin server or intermediate node) that has to serve a small set of nodes (respectively intermediate nodes or edge servers) in such a way that the throughput of the source is minimized and the streaming delay stays reasonable. We explore a solution based on rateless codes and data exchanges among the nodes. Since the latest rate-adaptive streaming protocols rely on a segmentation of streams into independent chunks, the use of *rateless codes* on every video chunk appears as an attractive idea. With rateless codes (*e.g.*, Raptor codes [12] and LT codes [9]), a video chunk of k information symbols can be recovered with high probability if slightly more than k encoded symbols are received. The idea is that the source applies rateless codes to every video chunk and delivers each created symbol to only one node, which then uses an application-level multicast tree to deliver this symbol to some other nodes.

1.1 Related Work

The first publications about live streaming in CDNs are ten years old (*e.g.*, [2, 4, 11]). Most of these publications emphasized the fan-out limitations of intermediate equipments. The focus of these papers was to build on top of large overlays a set of low-delay application-level multicast trees, subject to the constraints of the capacity of nodes. We revisit this objective through more recent delivery techniques, including rateless codes and video chunks. Moreover, we restrict our study to a specific local part of the whole overlay, where the bottleneck lies.

After five years without much academic activity, the delivery of a live stream in CDNs has become hot again with a series of recent publications [1,3,8]. These works consider the equipment bottleneck as well. They aim at creating delivery trees but their optimization purpose is generally to save bandwidth cost. Since we are within the CDN infrastructure, we neglect bandwidth cost.

Previous works related to rateless codes focus on peer-to-peer systems (*e.g.*, [6,14,15]). These works address problems related to scalability and management of churn. These issues are not major in CDN infrastructures, where the number of nodes is typically less than one hundred, and machines are rarely faulty.

1.2 Our Contributions

Our contributions are twofold:

- theoretical contribution: we aim at building a multi-tree structure that minimizes the source transmission rate. When we do not take into account the constraints of delay and packet loss, we give in §3.1 an exact algorithm (minimum source transmission rate). When we consider the joint height constraint of the delay and the reliability, we give in §3.2 an effective heuristic algorithm.

- practical contribution: our evaluation in §4 highlights that when the nodes can reserve an upload bandwidth that is larger than the video bit-rate, the source needs slightly more than the video bit-rate to transmit a video block to up to 25 nodes in a short delay and in a reliable way.

Our paper makes a first step toward addressing the problem of *equipment bottleneck* within CDN networks.

The rest of this paper is organized as follows. In §2, we present the system model and formulate the problem of minimizing the throughput for live streaming in CDNs. To this end, we propose two diffusion forest construction algorithms in §3. Then we evaluate the performance of our heuristic algorithm with simulations in §4. Finally, we conclude this paper and mention future work in §5.

2. SYSTEM MODEL

The source is the equipment that is the bottleneck of the CDN infrastructure. We suppose that the CDN is able to identify it. We denote it by s. This source is expected to deliver a flow of video chunks to some other equipments within the CDN networks. We denote by V this set of equipments ($|V| = n$), and we denote by $G = (V, E)$ the connected symmetric digraph modeling the small part of the whole CDN network that we consider in this paper. This is the network that is affected by the lack of upload capacity of the source. Our goal is to prevent the bottleneck to affect the whole CDN, thus we propose that these nodes cooperate in order to fix the bottleneck issue. We refer to this network as the *support network*. As usual, E is the set of links between nodes ($|E| = m$). The source s does not belong to V. An arc from u to v is denoted by (u, v). We assume that the support network is fully connected.

Every node $v \in V$ is associated with a *support capacity* (or capacity for short), denoted by c_v, which is the number of packets the node v can relay in the support network. It is important to understand that the equipments are not expected to re-transmit data to their sibling equipments in a hierarchical infrastructure like today's CDN networks. That is, intermediate nodes do usually not transmit data to intermediate nodes, and edge servers do not communicate with other edge servers. But there is not a lot of options for addressing the equipment bottleneck. We thus consider that the CDN has "reserved" a small fraction of the overall upload capacity in every equipment for the support network.

The *transmission delay* over the link (u, v) is d_{uv}. We assume that the transmission delay is a real value in $(0, 1]$, for example a fraction of the maximum Round-Trip Time (RTT) experienced in the CDN network.

The *transmission reliability* over the link (u, v), which corresponds to the probability that a packet sent by u is successfully received by v, is denoted by p_{uv}. It is also in $(0, 1]$. Note that the delay is additive while the reliability is multiplicative over arcs.

In the support network, we use a multi-tree structure for the diffusion of each video chunk. Each tree is the support for the transmission of *one* packet of encoded symbols from the source to nodes. To recover the video chunk, a node should receive at least K packets of encoded symbols. In other words, a node should belong to at least K trees. We illustrate our proposal in Figure 1. On the left, in the gray box, we represent the support network. On the right, we represent the four trees that allow each node to receive $K = 3$ packets of encoded symbols.

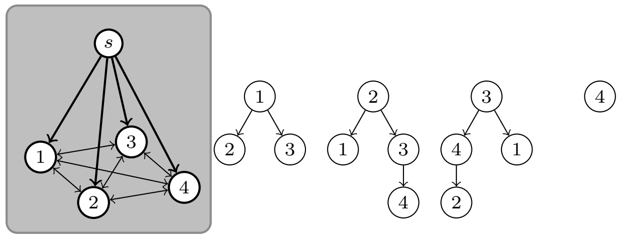


Figure 1: Multi-tree structure for the diffusion of each video chunk (four nodes and $K = 3$)

As the capacity of each node is limited, it may be impossible to use one tree to relay a packet to all nodes. In addition, to guarantee on-time delivery and reliability at the nodes, the end-to-end delay and transmission reliability on the path from the source to each node should be bounded to a reasonable value.

Our objective is to minimize the transmission rate for the source. The more trees are used in order to ensure that all nodes receive at least K packets, the higher is the source transmission rate. Thus, minimizing the source transmission rate is equivalent to minimizing the number of trees. In light of this, we formulate the video diffusion problem in the support network as a *Height-Bounded Spanning Forest problem with Capacity constraint* (HBSFC). The goal of HBSFC is to find a forest F with minimum cardinality such that

- each tree $T \subseteq G$ is rooted on a node, which directly receives a packet from s. For simplicity, we assume no packet loss between s and a root.
- for each node $v \in V$, the sum of its out-degrees in all trees of F is not greater than its capacity c_v. Let $deg_T^+(v)$ be the out-degree of v in $T \in F$,
$$\forall v \in V, \quad \sum_{T \in F} deg_T^+(v) \leq c_v \tag{1}$$
- each node $v \in V$ is spanned in at least K trees of F. Let $F(v)$ be the set of trees in which v is included. Then $|F(v)| \geq K, \forall v \in V$.
- for each tree $T \in F$, the transmission delay from the root to each node in T is no bigger than a given threshold $D_h > 0$. That is, the *delay height* of each tree is bounded by D_h.
- for each tree $T \in F$, each node $v \in T$ receives a packet from the root with a reliability no smaller than a reliability bound $P_h \in (0, 1]$. Let $SP_T(v)$ be the list of arcs in the path from the root to v in tree T. Then
$$\forall T \in F, \forall v \in T, \prod_{e \in SP_T(v)} p_e \geq P_h \tag{2}$$
By applying logarithms on both sides, we obtain:
$$\forall T \in F, \forall v \in T, \sum_{e \in SP_T(v)} \log(p_e) \geq \log(P_h) \tag{3}$$
which means that the *logarithmic reliability height* of each tree is bounded by $\log(P_h)$.

3. DIFFUSION FOREST CONSTRUCTION ALGORITHMS

We propose two *polynomial-time* algorithms to optimize the diffusion forest in two models: unbounded height model and bounded height model.

3.1 Unbounded Height Model

We first assume an unbounded height model: (i) the transmission reliability is $p_e = 1$ for all $e \in E$, and (ii) the delay threshold D_h satisfies $D_h \geq n - 1$, so that no tree will violate this constraint. Please refer to Algorithm 1 for the pseudocode of our algorithm. Note that notation $\overline{V_k}$ stands for $V \setminus V_k$ and (V_k, E_k) represents tree T_k. We use c'_v to denote the available capacity of a node v.

This algorithm consists of two phases. First, we construct K trees, and maximize the number of nodes that are spanned

Algorithm 1: Unbounded Height Model

```
Input : Complete graph $G = (V, E)$, capacity $c_v$ of
        $v \in V$, number of packets $K$
Output: A forest $F$
1  $k \leftarrow 1$;
2  $F \leftarrow \emptyset$;
3  for $v \in V$ do $c'_v = c_v$;
4  $V_c \leftarrow \{v : c'_v > 0, v \in V\}$;
5  while $k \leq K$ and $V_c \neq \emptyset$ do
6      find node $r \in V_c$ s.t. $c'_r = \max\{c'_v : v \in V_c\}$;
7      $V_k \leftarrow \{r\}$;
8      $E_k \leftarrow \emptyset$;
9      $T_k \leftarrow (V_k, E_k)$;
10     while $V_k \neq V$ and $V_c \neq \emptyset$ do
11         find node $u \in \overline{V_k}$ s.t. $c'_u = \max\{c'_v : v \in \overline{V_k}\}$;
12         pick any node $w \in V_k \cap V_c$;
13         $V_k \leftarrow V_k \cup \{u\}$;
14         $E_k \leftarrow E_k \cup \{(w, u)\}$;
15         $T_k \leftarrow (V_k, E_k)$;
16         $c'_w \leftarrow c'_w - 1$;
17         if $c'_w = 0$ then
18             $V_c \leftarrow V_c \setminus \{w\}$
19     $F \leftarrow F \cup \{T_k\}$;
20     $k \leftarrow k + 1$;
21 for $v \in V$ do
22     $k_v \leftarrow$ number of trees $v$ belongs to;
23     if $k_v < K$ then
24         add $(K - k_v)$ trees $(\{v\}, \emptyset)$ to $F$
```

in these trees. For each tree, the root r is selected as the node having the biggest available capacity. Then, we attach nodes to this tree starting from the nodes with biggest available capacity until either all nodes are spanned or no node has available capacity. Second, we take care of nodes that are not spanned in the K trees. As all nodes have exhausted their capacities, we span them by building *shallow trees* containing only one node. For example, the tree on the right in Fig. 1 is a shallow tree as it contains only the node 4.

THEOREM 1. *In the unbounded height model, the optimal forest has cardinality*
$$|F| = \begin{cases} K & \text{if } \sum_{v \in V} c_v \geq K(n-1) \\ K \times n - \sum_{v \in V} c_v & \text{otherwise} \end{cases} \tag{4}$$

PROOF. Let c_v^F be the sum of the out-degrees of node v in F. Then the number of nodes in F is equal to $\sum_{v \in V} c_v^F + |F|$. In an optimal solution, exactly $K \times n$ nodes belong to F. We have
$$|F| + \sum_{v \in V} c_v \geq |F| + \sum_{v \in V} c_v^F = K \times n \tag{5}$$
When $|F| = K$, the sum of capacities should thus verify:
$$\sum_{v \in V} c_v \geq K(n-1) \tag{6}$$
Otherwise, the total capacity is not enough to span $K \times n$ nodes in K trees. To minimize $|F|$, we should use all the nodal capacities, i.e., $\forall v \in V, c_v^F = c_v$. Thus, we obtain a forest cardinality of $|F| = K \times n - \sum_{v \in V} c_v$. $\square$

Theorem 2. *Algorithm 1 computes an optimal solution*

Proof. At step (5) of Algorithm 1, the first loop terminates when one of the following conditions is satisfied:

- All nodes are spanned K times in the K trees. Thus only K trees are required, $|F| = K$. Every tree in F contains n nodes, so exactly $K(n-1)$ out degree is needed. Thus we have $\sum_{v\in V} c_v \geq K(n-1)$.
- All nodes have exhausted their capacity ($V_c = \emptyset$) before finishing K trees. Let $\overline{k} \leq K$ be the number of constructed trees. These trees contain $\overline{k} + \sum_{v\in V} c_v$ nodes. As no node has available capacity, all remaining trees in the resulting forest are shallow trees, and $K \times n - (\overline{k} + \sum_{v\in V} c_v)$ shallow trees are needed. Thus,

$$|F| = \overline{k} + K \times n - (\overline{k} + \sum_{v\in V} c_v) = K \times n - \sum_{v\in V} c_v \tag{7}$$

In short, the forest built by Algorithm 1 has minimum cardinality and spans each node exactly K times. □

At each step of Algorithm 1, it takes $\mathcal{O}(n)$ time to find the node with the biggest capacity and there are at most $K \times n$ steps, thus the overall time complexity is $\mathcal{O}(Kn^2)$.

3.2 Bounded Height Model

We now take into account both the delay and the reliability constraints. We suppose that the delay and reliability are identical over all links, i.e., $\forall e \in E, d_e = d > 0, p_e = p < 1$. Let $h(T)$ be the height of T, the delay height constraint and the reliability constraint in equation (3) are thus simplified into one joint tree height constraint, i.e.

$$\forall T \in F, h(T) \leq H = \min\{\lfloor \frac{D_h}{d} \rfloor, \lfloor \frac{\log(P_h)}{\log(p)} \rfloor\} \tag{8}$$

This is to say that the number of hop counts from the root to each node is bounded by H in any tree. When $H = 0$, each tree contains only a node, and thus Kn trees are required. When H is bigger than $n-1$, the unbounded height algorithm that we previously presented can be used. For all other heights, we propose a time-efficient heuristic algorithm, detailed in Algorithm 2. We use $h(T_k)$ for the height of tree T_k and $h_{T_k}(w)$ for the depth of a node w in T_k. In the pseudocode, V_K stores the set of nodes not yet spanned in K trees, and V_k^+ stores the nodes via which a node can be added to T_k without violating the height and capacity constraint.

Following the same idea as in Algorithm 1, our algorithm consists of two phases. The first one exhausts node capacity and the second one creates additional shallow trees if necessary. For every tree, we select the root as the node having the biggest available capacity, then we attach other nodes to the tree in a breadth-first manner starting from the nodes having biggest capacity until either the height of the tree reaches $H-1$ or no more unattached node has capacity. Note that a node can be attached although it has already been spanned K times. At the end of each tree, we add the nodes that are still in V_K starting from the nodes with the smallest capacity until until either all nodes are spanned, or all capacities of nodes with height smaller than H are exhausted, i.e., $V_k^+ = \emptyset$. In the second phase, we build the remaining shallow trees if V_K is not empty. At most $K \times n$

Algorithm 2: Bounded height model

```
Input : Complete graph $G = (V, E)$, capacity $c_v$ of $v \in V$, number of packets $K$, and height $H$
Output: A height-bounded forest $F$
1  $k \leftarrow 1$;
2  $F \leftarrow \emptyset$;
3  $V_K \leftarrow V$;
4  for $v \in V$ do  $c'_v = c_v$;
5  $V_c \leftarrow \{v : c'_v > 0,\ v \in V\}$;
6  while $V_c \neq \emptyset$ and $V_K \neq \emptyset$ do
7      find node $r \in V_c$ s.t. $c'_r = \max\{c'_v : v \in V_c\}$;
8      $V_k \leftarrow \{r\}$;
9      $E_k \leftarrow \emptyset$;
10     $T_k \leftarrow (V_k, E_k)$;
11     if $r \in V_K$ and $r$ is spanned $K$ times then
12         $V_K \leftarrow V_K \setminus \{r\}$;
13     $V_k^+ \leftarrow \{v : v \in V_k \cap V_c$ and $h_{T_k}(v) < H\}$;
14     while $\overline{V_k} \cap V_K \neq \emptyset$ and $V_k^+ \neq \emptyset$ do
15         find $w \in V_k^+$ s.t.
           $h_{T_k}(w) = \min\{h_{T_k}(v) : v \in V_k^+\}$;
16         if $h_{T_k}(w) < H - 1$ and $\overline{V_k} \cap V_c \neq \emptyset$ then
17             find $u \in \overline{V_k}$ s.t. $c'_u = \max\{c'_v : v \in \overline{V_k}\}$;
18         else
19             find $u \in \overline{V_k} \cap V_K$ s.t.
               $c'_u = \min\{c'_v : v \in \overline{V_k} \cap V_K\}$;
20         $V_k \leftarrow V_k \cup \{u\}$;
21         $E_k \leftarrow E_k \cup \{(w, u)\}$;
22         $T_k \leftarrow (V_k, E_k)$;
23         $c'_w \leftarrow c'_w - 1$;
24         if $c'_w = 0$ then
25             $V_c \leftarrow V_c \setminus \{w\}$;
26         $V_k^+ \leftarrow \{v : v \in V_k \cap V_c$ and $h_{T_k}(v) < H\}$;
27         if $u \in V_K$ and $u$ is spanned $K$ times then
28             $V_K \leftarrow V_K \setminus \{u\}$;
29     $F \leftarrow F \cup \{T_k\}$;
30     $k \leftarrow k + 1$;
31 for $v \in V_K$ do
32     $k_v \leftarrow$ the number of trees $v$ belongs to;
33     add $K - k_v$ trees $(\{v\}, \emptyset)$ to $F$
```

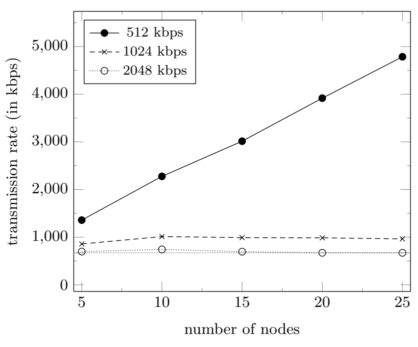
trees may be computed and each tree contains at most n nodes. It also takes $O(n)$ time to find the node with the maximum or minimum capacity. Thus the time complexity is $O(Kn^3)$.

4. SIMULATIONS

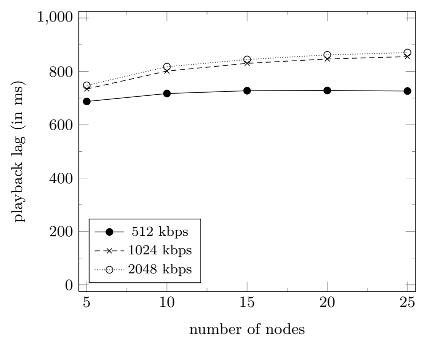
We use the ns-2 network simulator to evaluate the performance of our heuristic algorithm (Algorithm 2).

4.1 Video and Rateless Code Settings

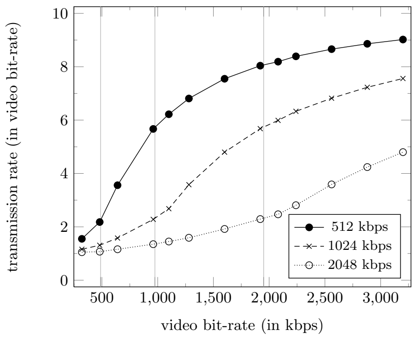
The video was compressed with the H.264 coder at various bitrates ranging from 320 kbps to 3.2 Mbps. The resulting bitstream was partitioned into chunks where each chunk corresponded to one Group of Pictures (GOP). Each chunk had a playback duration of 0.5 s. The source applies rateless coding on each chunk and sends the encoded symbols in successive UDP packets of size 1,000 bytes. The size of



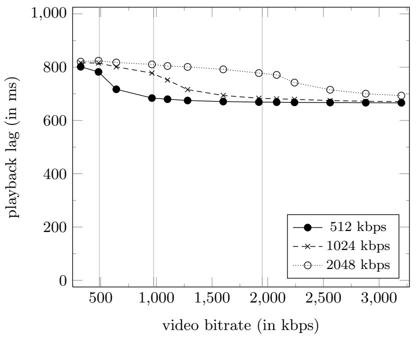
(a) source rate vs. number of nodes



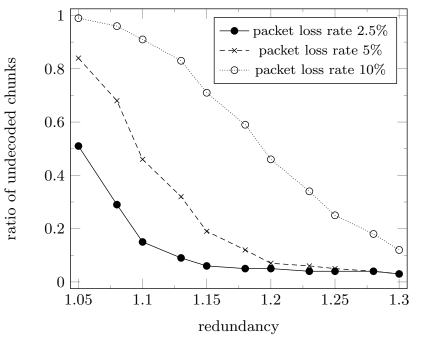
(b) decoding lag vs. number of nodes



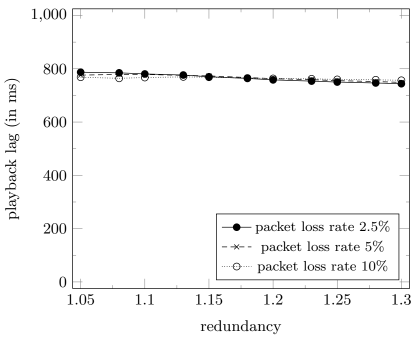
(c) source rate vs. video bit-rate



(d) decoding lag vs. video bit-rate



(e) outage rate vs redundancy



(f) decoding lag vs. redundancy

Figure 2: Results of our evaluation

an encoded symbol was one byte. For rateless coding, the Raptor code model proposed in [10] was used. With this model, a redundancy of 5%, gives a very high probability of successful decoding [10]. Thus, for a default video bitrate of 640 kbps, a node needs about 672 kbps to recover the original chunk.

4.2 Nodes Settings

The number of nodes varied from 5 to 25 (by default 10). Since there is no existing solution like the one we propose, it is hard to set the support capacity of the node. Our goal is to explore various capacity distributions and to observe the impact on the performance. Thus, our simulations can guide CDNs to set the support capacity. We used a log-normal distribution. The first parameter is the mean upload capacity, which was chosen in the set {512 kbps, 1,024 kbps, 2,048 kbps}. The second parameter is σ, which we abusively call *capacity heterogeneity*. We used three values for σ: $\{0.1, 0.8, 1.5\}$. The first value $\sigma = 0.1$ corresponds to a homogeneous configuration where the capacity of all nodes is close to the mean. At the other end, the value $\sigma = 1.5$ generates heterogeneous configurations where a few nodes have a high capacity while the remaining ones have a low capacity. As we have observed a low impact of the capacity heterogeneity, we do not present curves with distinct heterogeneities.

4.3 Network and Algorithm Settings

The core network of the CDN is a homogeneous network where most equipments and links are standard. Thus, we assume that both RTT and the packet loss probability are identical on every link (respectively set to 50 ms, and, for the packet loss probability to zero in Figs. 2(a) to 2(d) and 0.025, 0.05, or 0.1 in Fig. 2(e) and 2(f)). The Raptor code redundancy was set to 5% in Figs. 2(a) to 2(d). We used three different height bounds, $H \in \{1, 2, 3\}$. As for the capacity heterogeneity, we do not present results for each tree height bound separately because this parameter had a low impact on the overall performance. Therefore, every point on a curve corresponds to the average value of ten runs for three different tree height bounds and three different capacity heterogeneities.

4.4 Discussion

Figs. 2(a) and 2(c) show the required transmission rate at the source to guarantee that all nodes are able to decode the video chunk for various numbers of nodes and video bit-rates, respectively. Figs. 2(b) and 2(d) show the average lag between the time at which the source sends the first packet of a video chunk and the time at which a node is able to decode the video chunk again for various numbers of nodes and video bit-rates, respectively. Gray vertical lines in Figs. 2(c) and 2(d) indicate the video bit-rate for which the support network is just-provisioned for our different mean upload capacities. In the following, we use these four figures to highlight that our proposal has two distinct behaviors regarding the provisioning of the support network.

When the support network is under-provisioned, upload resources are exhausted before all nodes are spanned K times, so the source has to compensate for the lack of upload capacity by directly sending the last packets of the video chunk in shallow trees. On one hand, these shallow trees are costly in terms of source transmission rate because each one transmits one packet to only one node. On the other hand, as no relay is necessary, the decoding lag is lower. The more under-provisioned is the support network, the larger is the number of shallow trees in the forest. Therefore in the extreme case (video bit-rate 3.2 Mbps for mean capacity 512 kbps), the playback delay is stable around the minimum transmission delay, and the source transmission rate is roughly equal to the number of nodes times the video bit-rate.

When the support network is over-provisioned, the source does not need to build shallow trees. Our algorithm ensures that the forest contains compact and well-balanced trees. The decoding lag is less than 400 milliseconds, even when 25 nodes should be served. Our algorithm also succeeds in fully utilizing the resources of the nodes, as demonstrated by the source transmission rate, which is stable around the optimal lower bound in Fig. 2(a) for a mean capacity 2,048 kbps and never requires the transmission of more than 2.5 times the video bit-rate in every over-provisioned configuration. Our ability to serve a large number of nodes in a short delay makes the case for our proposal.

We then evaluate the performances of our algorithm when UDP packets can be lost during transmission. Rateless codes allow the diffusion of a potentially infinite number of distinct symbols until all nodes can decode the chunk. In our context, we fix a redundancy index, which indicates the bit-rate of the encoded stream including the redundancy in comparison to the video bit-rate (*e.g.*, a redundancy of 1.05 for a video bit-rate of 1,000 kbps means a stream bit-rate of 1,050 kbps). Fig. 2(e) shows the *outage rate*, which is the ratio of chunks that are not decoded to the overall number of transmitted chunks. Fig. 2(f) shows the decoding lag of the successfully decoded chunks.

Since the packets may be relayed many times before reaching a node, packet loss has a dramatic impact on successful chunk decoding. With low redundancy (less than 1.10), the outage rate is intolerable when the network is faulty (almost no chunk successfully decoded when 10% of transmissions are faulty). However, the benefits of using rateless codes become clearer for stream bit-rates that include a reasonable redundancy (between 1.15 and 1.20). Here, the outage rate falls below 0.2 in the most realistic scenarios. Finally, as shown in Fig. 2(f), neither the redundancy nor packet loss affect the decoding lag. When the redundancy increases, the number of shallow trees increases, and the decoding lag paradoxically decreases.

5. CONCLUSION AND FUTURE WORK

In this paper, we try to prevent one equipment (source) to throttle a whole CDN infrastructure in the context of live stream delivery. We propose the notion of support network where the equipments that receive the video chunks from the source cooperate in order to compensate for its missing capacity. We study one solution for this support network, which consists in leveraging on rateless codes. We modeled the problem as finding a diffusion forest with guaranteed end-to-end delay and bounded packet loss probability such that the total number of trees in the forest is minimized. We presented an optimal solution when both the delay and packet loss rate are not bounded. When the delay and the reliability are bounded and identical over all links, we proposed a heuristic algorithm.

The numerical results demonstrate the performance of the heuristic algorithm in diverse system configurations. Future work will include a real implementation in a real CDN infrastructure in order to validate the practical interest of our solution on the overall CDN.

Acknowledgment

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7, 2007-2013) under the grant agreement no. ICT-248175 (CNG project).

6. REFERENCES

[1] M. Adler, R. K. Sitaraman, and H. Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18):4007–4020, 2011.

[2] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. J. Wright. Minimizing delivery cost in scalable streaming content distribution systems. *IEEE Trans. Multimedia*, 6(2):356–365, 2004.

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

[4] K. Andreev, B. M. Maggs, A. Meyerson, and R. K. Sitaraman. Designing overlay multicast networks for streaming. In *Proc. ACM SPAA*, pages 149–158, 2003.

[5] R. Gibbs. A new approach to publishing and caching video. Technical report, Alcatel-Lucent, Jan. 2012.

[6] M. Grangetto, R. Gaeta, and M. Sereno. Rateless codes network coding for simple and efficient P2P video streaming. In *Proc. of ICME*, pages 1500–1503, Jul. 2009.

[7] S. Higginbotham. Smart TVs cause a net neutrality debate in S. Korea. Giga OM, Feb. 2012.

[8] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. *Signal Processing: Image Communication*, 27(4):288–311, 2012.

[9] M. Luby. LT Codes. In *Proc. of FOCS*, pages 271–280, 2002.

[10] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson. Reliable multimedia download delivery in cellular broadcast networks. *IEEE Trans. Broadcast.*, 53(1):235–246, 2007.

[11] J. Ni and D. H. K. Tsang. Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. *IEEE Commun. Mag.*, 43(5):98–105, May 2005.

[12] A. Shokrollahi. Raptor codes. *IEEE Trans. Inf. Theory*, 52(6):2551–2567, 2006.

[13] T. Siglin. Super Bowl Streaming Fail. Streaming Media, Feb. 2012.

[14] N. Thomos and P. Frossard. Network coding of rateless video in streaming overlays. *IEEE Trans. Circuits Syst. Video Technol.*, 20(12):1834–1847, 2010.

[15] C. Wu and B. Li. rstream: Resilient and optimal peer-to-peer streaming with rateless codes. *IEEE Trans. Parallel Distrib. Syst.*, 19(1):77–92, 2008.