Joint Content Distribution and Traffic Engineering of Adaptive Videos in Telco-CDNs

Khaled Diab
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

***Abstract*—Telco-CDNs refer to content distribution networks deployed and managed by Internet Service Providers (ISPs). They are getting popular among major ISPs because they offer new revenue streams and have the potential of providing better performance compared to traditional CDNs. Managing telco-CDNs is, however, a complex problem, because it requires jointly managing the network resources (links and switches) and the caching resources (processing and storage capacities), while supporting the adaptive nature and skewed popularity of multimedia content. To address this problem, we present a new algorithm called CAD (Cooperative Active Distribution), which strives to serve as much as possible of the requested multimedia objects within the ISP while carefully engineering the traffic paths through the network. This is achieved by enabling the cooperation among caches within the ISP not only to serve various representations of multimedia objects, but also to create them on demand using the available processing capacity of caches. We have implemented CAD and evaluated it on top of a network emulator that runs deployment code and processes real traffic. Using an actual ISP topology, our experimental results show that CAD achieves substantial performance improvements compared to the closest work in the literature, e.g., up to 64% reduction in the total inter-domain traffic.**

I. Introduction

The amount of multimedia traffic distributed over the Internet has been increasing at high rates in the past several years [1]. Content Delivery Networks (CDNs) have played a critical role in supporting this increasing demand. Current CDNs, e.g., [2], replicate content at different caching locations and direct users to the closet/best location based on various factors such as geographical distance, end-to-end latency, and traffic load at different locations.

To improve user-perceived quality, current CDNs often infer network conditions inside ISPs through complex measurement methods [2], [3]. However, CDNs cannot control the traffic flows and what paths they take inside ISP networks, which ultimately carry the actual traffic. CDNs may not even precisely know the underlying network topology and the current traffic situation on links and switches in the ISP network.[1] Thus, decisions made by CDNs may negatively impact the load on various links in the ISP networks [7], especially links between different network domains (called inter-ISP links), which are costly [8]. This may trigger ISPs to adjust traffic routing inside their networks, which in turn, could impact the performance of CDNs. To reduce the mis-match between the goals of CDNs and ISPs, multiple ISP-CDN collaboration models have been proposed, from defining interfaces that ISP and CDN can use to share information [9], to allowing content providers to deploy caches within ISPs (e.g., Netflix OpenConnect [10]). While useful, these approaches can only provide partial solutions. For example, OpenConnect caches [10] can provide local access to popular content within the ISP, but they cannot control the network paths taken to carry the multimedia sessions.

The difficulties of enabling ISP-CDN collaboration and the potential business opportunities motivated major ISPs, e.g., AT&T, to deploy and manage CDNs inside their networks [11], [12]. Such CDNs are referred to as telco-CDNs. Telco-CDNs offer unprecedented opportunities for optimizing the delivery of multimedia content, because not only can they *jointly* leverage the up-to-date information about the underlying ISP network *and* the characteristics of the multimedia objects, but they can also choose the appropriate paths for different sessions and configure the network to enforce these decisions. Managing telco-CDNs is, however, a complex task, because it requires concurrently managing resources at the network layer (traffic engineering (TE)) and system layer (processing and storage capacities of caches), while supporting the adaptive nature and skewed popularity of multimedia content.

In this paper, we propose a comprehensive solution to efficiently solve the resource management problem in the emerging telco-CDNs, which we refer to as CAD (short for Cooperative Active Distribution). CAD has two goals: (i) reducing the cost incurred by ISPs, and (ii) improving the quality of multimedia sessions by reducing end-to-end latency. CAD achieves these goals by serving as much as possible of the multimedia traffic locally within the ISP, while carefully engineering the traffic paths to avoid creating bottlenecks in the network. CAD can significantly improve the performance of multimedia streaming systems in scenarios where ISPs and content providers cooperate (e.g., Netflix OpenConnect [10]), and when ISPs own their streaming services such as AT&T U-Verse and Bell Canada. Both scenarios are increasingly seeing wider adoption and deployment in practice.

We present multiple novel ideas in CAD, and we believe that

[1] Although there are tools for inferring ISP topologies [4] and estimating network traffic [5], [6], they are, in general, quite expensive and cannot be used for obtaining accurate information about large-scale ISPs in *real-time*.

each of them can be beneficial in its own right. For example, we consider the fine-grained caching of multimedia objects. In particular, in current streaming systems, e.g., Netflix, a multimedia object is transcoded into multiple forms of qualities/resolutions/formats (referred to as representations) in order to support a wide-range of users with heterogeneous devices and dynamic network conditions. Instead of considering the popularity at the level of multimedia objects, we account for the popularity of individual representations of each object. This is important as some representations are requested much less than others, e.g., a representation customized for uncommon display or rare network condition. In addition, major ISPs deploy storage infrastructures and general-purpose multi-core servers inside their networks [3]. We leverage this processing power in creating some representations on demand, instead of pre-creating and storing all of them. This can save inter-ISP bandwidth, as a requested representation that is not locally stored can be created by transcoding another representation instead of having to fetch it from outside the ISP. This, however, needs to be done carefully so that the additional delay introduced by the on-demand creation of representations does not exceed a predefined threshold. Another novel aspect of our solution is that it enables the cooperation among caches within the ISP not only to serve representations, but also to create them on demand. Finally, we consider the active creation and cooperative serving of representations in the computation of the traffic paths in the network.

We have implemented CAD and evaluated it on top of Mininet [13], which processes and carries real traffic. We implemented caching servers that stream multimedia content using the widely-used DASH protocol [14]. We used virtual switches to realize the TE aspects of CAD, which are enforced using SDN (Software-defined Networking) rules. We used an actual ISP topology and conducted extensive experiments to assess the performance of CAD and compare it against the closest work in the literature [15]. Our results show substantial improvements across multiple performance metrics. For example, CAD achieves up to 64% reduction in the inter-domain traffic and 14% improvement in the buffer occupancy (i.e., number of buffered segments) at clients compared to the algorithm in [15]. The buffer occupancy indicates a better quality of experience (QoE) for users in adaptive streaming [16]. Our results also show that CAD does not overload the intra-domain links. This ensures the stability of the inter-domain routing policies employed by the ISP [17].

II. Related Work

We summarize the relevant works in the following.

ISP–CDN Collaboration: Most current CDNs are run independently from ISPs that own and operate the underlying networks. Thus, such CDNs may not have access to accurate and current information about the network topology and traffic conditions [2], [9]. This may result in higher costs and lower performance for both the CDN and ISP. To overcome these problems, works such as NetPaaS [9] enable ISP-CDN collaboration. These works focus on *information sharing*. In contrast, CAD *jointly* solves the content distribution and TE problems, which is a more challenging problem due to the heterogeneity of content and its popularity.

Major content providers impose significant load on ISP networks. For example, IX Australia reported peering traffic increase by 100% for some ISPs in Australia because of the high demand of streaming Netflix videos.[2] To partially mitigate this high load, Netflix started installing caches inside ISPs and deploying peering links to Internet eXchange Points (IXPs). Similarly, the Streaming Video Alliance has recently started trials to bring caching servers to ISP networks [18]. Unlike CAD, most of these caches work independently, and they cannot control the network paths taken by streaming sessions within the ISP.

Joint Caching and TE: Xie et al. [19] formulated the TE-guided collaborative caching (TECC) problem for in-network caching systems. TECC only calculates the amount of traffic for every source-destination pair along a direct path. CAD, however, supports multipath routing, and calculates network paths for every source-destination-video tuple. Unlike TECC, CAD utilizes the processing power at caching sites and the relationship between representations to create some of them on demand. Jiang et al. [20] developed a collaboration mechanism between ISPs and content providers to coordinate the TE and server selection, assuming that the content is already placed at caching servers.

Telco-CDNs: Telco-CDNs share the vision of the recent proposal in [21], which allows content providers to reach inside ISPs and control their own traffic flows through network virtualization to optimize the user experience. The work in [21], however, only outlines the vision and its potential, but it does not provide algorithmic solutions to manage caching resources and network traffic. Li and Simon [22] proposed a push-based approach to manage telco-CDNs, but it does not manage the network resources. Zhou et al. [23] proposed an algorithm to maximize the number of live streaming sessions served with the lowest bandwidth in telco-CDNs. In contrast, CAD manages telco-CDNs to support on-demand streaming.

Online Transcoding Systems: Cloud transcoding systems have emerged to create video representations on demand. Gao et al. [24] jointly scheduled transcoding tasks and provisioned computation resources to maximize revenue and satisfy user requirements. Krishnappa et al. [25] proposed an online transcoding system for CDNs. Ma et al. [26] built a transcoding time estimator and integrated it with task scheduler to reduce transcoding completion time. These systems do not consider cooperation between servers nor do they support TE.

In summary, unlike previous systems from industry and academia, CAD *simultaneously* considers: (1) the joint optimization of caching resources and the engineering of traffic paths in the network to serve adaptive multimedia content, (2) the available processing capacity at caches to create video representations on-demand, and (3) the cooperation among caches to serve and create video representations on demand.

[2]http://bit.ly/2htgP4b

III. System Models and Problem Definition

A. Multimedia Content Model

We consider multimedia objects served using adaptive streaming protocols, such as DASH [14], to support heterogeneous clients and accommodate dynamic network conditions. In this model, a multimedia object (e.g., video) is encoded in multiple *representations* based on several factors, including available bandwidth, screen resolution, screen type, and media player. All representations are created from the original copy, which is referred to as the *master representation*. Each representation is divided into equal-length *segments* in the order of one to a few seconds. In most current CDNs, all representations are pre-created and stored before starting the distribution process. In the proposed telco-CDN, we create some of the representations on demand inside the ISP, which can be done given the target environment for CAD is telco-CDN where the content and the network resources are managed/owned by the same entity (or cooperating entities).

In addition, representations of the same multimedia objects may have different popularity. For example, a representation customized for uncommon network conditions will see fewer client requests compared to a representation customized for typical network conditions. CAD considers the popularity of individual representations, which provides more room for optimizations compared to the current model of considering the popularity of a multimedia object without accounting for the relative importance of its various representations. The popularity of a representation is defined in terms of the number of requests for that representation during the previous periods.

In the current and widely deployed DASH protocol, each client dynamically chooses the most suitable representation based on the rendering device characteristics and network conditions. CAD does not change the client adaptation method, nor does it require any modifications to the DASH protocol. Rather, CAD efficiently manages the telco-CDN such that clients obtain the requested objects with shorter latency from local caches through carefully-engineered network paths. This may indirectly enable the client adaptation method to request better quality representations and improve the QoE for users.

B. Telco-CDN Model

A simplified view of the considered telco-CDN is illustrated in Figure 1, where an ISP deploys caching servers at different sites to serve multimedia objects to clients with short latency, while minimizing the traffic load on inter-ISP links. The telco-CDN is managed by the proposed CAD (Cooperative Active Distribution) algorithm, which runs on a server within the ISP. The multimedia objects are provided by content providers interested in the delivery services offered by the telco-CDN. Content providers encode each multimedia object in multiple representations. The expected demand for each representation is estimated using predictive techniques such as [27], [28], and can be performed either by the telco-CDN or the content provider. We do not address demand estimation methods in this paper; our proposed CAD algorithm can work with any demand estimation method. We design CAD to augment streaming systems where the demand of multimedia objects can be estimated. For example, Netflix and Hulu may estimate content demands based on its release cycles. Note that these systems are increasingly attracting viewership, and they have agreements with major ISPs. Extending CAD to work in user-generated video streaming systems is left as a future work.

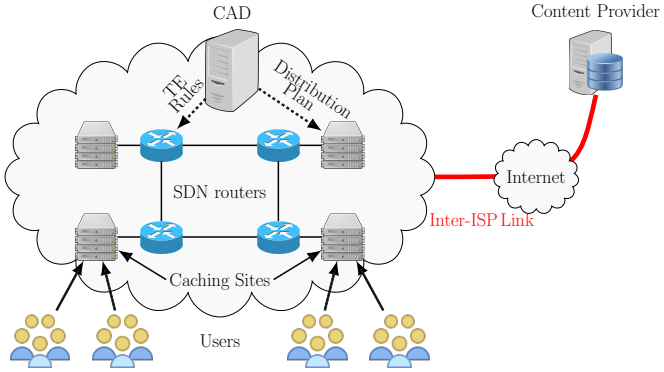


Fig. 1: A simplified view of the proposed CAD to manage telco-CDNs. CAD reduces the load on inter-ISP links and shortens the latency perceived by clients.

The design of CAD is motivated by the deployment of microdatacenters by major ISPs [3]. Caches in CAD are active and cooperative. They are active because they can create video representations using their own processing capacity, provided that the time to create these representations does not exceed a predefined threshold latency L seconds. The cooperative aspect of CAD allows one cache to fetch a requested representation from another, provided that appropriate network paths can be found to support the requested session. If a representation cannot be served from caches within the ISP, it will be requested from the content provider. This is all done transparently to end users.

C. Problem Definition

We consider a multi-region ISP operating a telco-CDN, which is modeled as a graph $(\mathbb{S}, \mathbb{E})$, where $\mathbb{S}$ represents caching sites and $\mathbb{E}$ represents links between the sites. Each caching site $s \in \mathbb{S}$ has one or more servers with aggregate storage capacity $b[s]$ bits and processing capacity $p[s]$ cycles/sec. Each link $e \in \mathbb{E}$ has a capacity of $c[e]$ bits/sec. The set of multimedia objects (videos) is denoted by $\mathbb{V}$, where each video $v \in \mathbb{V}$ has up to R representations. The popularity (or demand) of a video representation r at caching site s is denoted by $d[s][r]$ bits/sec.

The problem we address in this paper is to serve user requests from the telco-CDN by allocating the storage and processing resources at each caching site to the different video representations as well as directing the traffic flows of the video sessions through the ISP network such that the cost incurred by the telco-CDN is minimized (in terms of the amount of inter-ISP traffic) and the user-perceived quality is optimized (in terms of end-to-end latency). We divide this

problem into two sub problems: creating distribution plan (referred to as DP) and deciding the network paths for the video sessions (referred to as TE). The DP sub-problem determines for each caching site which video representations to store locally so that they can readily be served to clients near to that site, and for the remaining video representations which ones to: (i) create on demand, (ii) fetch from another caching site within the ISP, or (iii) fetch from the origin server. The TE sub-problem computes network paths through the ISP to minimize the maximum link utilization (MLU) inside the ISP network. With few modifications to the proposed solution, other TE objectives can be plugged into the proposed system.

IV. Proposed Solution

A. Overview

The proposed CAD algorithm *jointly* solves the DP and TE sub-problems. At high level, CAD runs periodically on a server inside the ISP. The period is a configurable parameter and can range from tens of minutes to several hours. In each period, CAD uses information about the popularity of various video representations and available resources inside the ISP (CPU cycles, storage, and available bandwidth on different links) to compute the distribution plan for all video representations. The distribution plan specifies for each caching site which video representation to store, create on demand, fetch from other caching sites, or fetch from the origin server. During the computation of the distribution plan, CAD considers the available bandwidth in the calculation and it specifies the network paths within the ISP to be taken to serve each video session, that is, it also solves the TE sub-problem while solving the DP sub-problem.

CAD produces two outputs: distribution plan and traffic engineering rules. The distribution plan is sent to the caching sites so that each server knows what to do upon receiving a user request. The traffic engineering rules are SDN rules that are sent to routers in the ISP network so that they can enforce the decisions made by CAD. The actual video sessions are created only when users submit requests. When a user submits a request for a video representation, the request is directed to the caching site nearest to the user, using methods commonly used in current CDNs such as DNS re-direction. Using the pre-computed distribution plan, the caching site can handle the user request locally, from other caches, or from origin server.

Jointly computing the distribution plan and TE rules of CAD is challenging. Specifically, representations are heterogeneous in terms of popularity, and storage and processing requirements. For example, creating a popular representation on-demand that has low processing requirements may result in service delays. In addition, cooperative and active distribution results in traffic inside the ISP network that needs to be routed efficiently. Moreover, single path routing may result in congestion at links inside the network. Finally, CAD should compute the distribution plan and TE rules in a timely manner.

We introduce multiple ideas to address these challenges. CAD computes for each representation a *value* that captures its importance based on its popularity, storage and processing requirements. Given the demand of a representation that is fetched or created on-demand, we create a weighted sub-network that spans the destination (i.e., demand site) and the sources (i.e., sites where the representation or its master copy are stored). Then, we transform the TE problem of a fetched or created on-demand representation to a min-cost flow problem in this sub-network. This allows CAD to support multi-path routing based on link usage to balance the total traffic across all links, and reduces the running time of the proposed algorithm. We note that CAD does not exceed the storage, processing and network resources of the telco-CDN.

Algorithm 1 CAD: Calculate $\mathcal{D}$ and $\mathcal{T}$

Input: $\mathbb{T}$: ISP network topology
Input: $d[s][r]$: demand for representation r at caching site s
Output: $\mathcal{D}$: distribution plan
Output: $\mathcal{T}$: traffic engineering rules
Initialization:
1. Calculate rep. value per site, and add it to $V[s]$
2. Sort all representations in $V[s]$ by their values, $\forall s \in \mathbb{S}$
3. Iterate over all sites $s \in \mathbb{S}$ and representations in $V[s]$
   If size($V[s][i]$) $\leq b[s]$:
   $\mathcal{D}[s][i]$.status = `stored`,
   sites[r].append(s),
   $b[s]$ -= size(r); last_stored(s) = i
4. $\gamma_{\text{current}} = 0$ // Current MLU is set to zero
5. $F = \{\}$ // Aggregate traffic map

```
for (s ∈ 𝕊) do
    for (i =last_stored(s)+1 ; i < rep_count; i++) do
        r = V[s][i]
        Construct sub-network 𝕋'
        f = FindPaths(s, d[s][r], 𝕋', sites[r])
        c = CreateRepAt(s, r)
        decision = min(f, c) // either f or c based on MLU
        if (decision) then
            γ_current = decision.γ
            UpdateDPandTE(r, decision, F)
T = CalcTERules(F)
return D and T
```

B. Details of CAD

A high-level pseudo code of the proposed CAD algorithm is shown in Algorithm 1. CAD computes and returns two outputs: $\mathcal{D}$ and $\mathcal{T}$. $\mathcal{D}$ is a data structure with a record $\mathcal{D}[s]$ for each caching site $s \in \mathbb{S}$. $\mathcal{D}[s]$ has one entry for each video representation r, which contains two fields: *status* and *sites*. The *status* field indicates whether the caching site s should store, create on demand, fetch from other caching sites, or fetch from the origin server the video representation r when a client submits a request for that representation. The *sites* field contains a list of caching sites to obtain a representation or its master copy from in case the *status* of that representation is set to fetch from other sites. The second output of CAD is $\mathcal{T}$, which is a list of traffic engineering rules for routers in the ISP network. Each rule is in the

form of $\langle RID, src, dst, iport, oports, weights\rangle$, where RID indicates the ID of the router to execute this rule. A TE rule specifies that the traffic flow coming from the src address on input port $iport$ and going to the dst address should be forwarded on the output ports $oports$ with different ratios specified by the $weights$. In the initialization stage, the status of all representations is set to fetch from the origin server, and the TE rules are set to the default ones used by the ISP (e.g., shortest paths computed by the intra-domain routing protocol).

CAD maintains multiple data structures, including 2-dimensional matrix V, list sites[r] for each representation r, and the aggregate traffic map F. For every caching site $s \in \mathbb{S}$, $V[s]$ is a list that sorts all video representations descendingly based on their values. The value of a representation r at caching site s is given by $d[s][r] \times \text{cpu}(r)/\text{size}(r)$. This value determines the priority of storing the representation at a caching site such that a popular small-size representation with high processing requirement has higher priority to be stored. The list sites[r] maintains the caching sites that r is stored at. The aggregate traffic map F contains the traffic value for every src-dst pair. Specifically, if $F[src, dst][i, j] = f_{(i,j)}$, then the aggregate traffic from router i to router j for demands from src to dst is $f_{(i,j)}$ units of traffic.

As shown in Algorithm 1, CAD starts with an initialization stage where all representations are sorted based on their values and the result is stored in $V[s]$. In addition, for every site $s \in \mathbb{S}$, CAD sets $\mathcal{D}[s]$ to store as many representations as the storage capacity $b[s]$ allows, and keeps an index of the last stored representation in last_stored(s) variable. CAD also updates the list sites[r] for each stored representation r. The current MLU γ_{current} is set to zero.

Calculating the distribution plan: After initialization, CAD attempts to update $\mathcal{D}$ for every representation at every site to either `fetch` or `create` and calculate $\mathcal{T}$ accordingly in Lines 1–10. For every site s, the algorithm iterates over all representations starting from the index last_stored(s)+1. In every iteration, the algorithm calls two functions: FINDPATHS and CREATEREPAT. Each function returns caching sites, network paths and the resulting maximum link utilization γ. A returned γ value of ∞ means no solution was found. The FINDPATHS function requires creating a sub-network $\mathbb{T}'$ (Line 4). It attempts to find TE paths that accommodate the demand of r from the caching sites stored in sites[r] to the local site s. The CREATEREPAT function checks if r can be created locally. After the two functions return, CAD chooses the option that minimizes the maximum link utilization γ in Line 7. CAD then updates the current MLU value γ_{current}, and calls UPDATEDPANDTE to update $\mathcal{D}$ and F.

First, the function UPDATEDPANDTE updates the corresponding $\mathcal{D}[s]$ status to either `fetched` or `created` based on the decision variable, and sets the caching sites. Second, based on the returned network paths, it aggregates the traffic between every two routers per src-dst pair in the F map. It also decreases the available bandwidth for every link. Third, if r is to be created on-demand at caching site s, it decreases the available processing capacity $p[s]$.

We provide more details on the FINDPATHS and CREATEREPAT functions. The goal of FINDPATHS is to balance the traffic across all links in the network given a demand value $d[s][r]$ and stored representations at sites[r]. To achieve this TE goal, we transform the TE problem to the min-cost flow problem in the sub-network $\mathbb{T}'$ as follows. We first construct a sub-network $\mathbb{T}' = (\mathbb{S}', \mathbb{E}')$ with vertexes and edges contained in all paths from sites[r] to s. Only edges (links) with remaining capacity greater than zero are included. The cost of every edge e in the sub-network is calculated as $\omega'_e = d[s][r]/c[e]$. Then, the costs of inter-ISP links are increased to double of their value in order to reduce the amount of inter-ISP traffic, because paths are selected based on lower costs. Finally, we solve the min-cost flow problem (MCF) using the cost-scaling algorithm in [29]. The algorithm returns a set of sites that will transmit the representations, a set of links with their traffic values, and the resulting maximum link utilization γ. To calculate γ, we iterate over all links in the output set of links, divide the total traffic for every link by its capacity, and choose the maximum value. We note that the cost-scaling algorithm requires integer edge costs. To meet this requirement, we multiply all costs by a relatively large integer (e.g., 100) and round up all costs to the nearest integer values.

The function CREATEREPAT checks if a representation r can be created on-demand locally within the allowed delay L seconds. First, it calculates the required CPU cycles to process $d[s][r]$. To calculate the cycles, it computes the segments count of r to be requested per second at caching site s. Then, it multiplies the processing requirement per segment by the segments count. If the required cycles are less than the available CPU resources in L seconds (i.e., $p[s] \times L$), the algorithm checks if the master copy is stored locally, and returns the site s as the a caching site. Since the master copy is stored locally, the MLU is not affected and we return the current MLU. If the master copy is not stored locally, the algorithm checks if the network can accommodate fetching it from sites[r.master]. It then calls FINDPATHS to find TE paths. If the network can fetch r.master, then the corresponding sites, links and MLU are returned.

Calculating the TE rules: CAD aggregates the traffic between every two routers per src-dst pair using the aggregate traffic map F. Then it calls CALCTERULES in Line 11 to find the TE rules. CALCTERULES takes F as an input and calculates the output port weights for every router per src-dst pair in the TE paths. To do so, CALCTERULES iterates over all src-dst pairs in F. For a router i, it finds all F entries where router i has output flows. To find the weight of port n that connects i to j, we divide the i, j output flow value over the sum of all output flow values for router i. That is $\mathcal{T}[src, dst][i][n] = F[src, dst][i, j]/\sum_k F[src, dst][i, k]$.

The computed TE rules is abstract enough to be used by various TE technologies such as OpenFlow [30], or Segment Routing [31] that does not require additional state at routers.

Time Complexity: CAD terminates in polynomial time in the order of $\mathcal{O}(N \log(D))$, where N is the total number of representations and D is the maximum demand.

V. Evaluation

A. Implementation

We implemented all components of the proposed CAD algorithm, from computing distribution plans and traffic engineering rules, to executing the distribution plans by caching sites and enforcing the rules by the ISP routers in real time. Our implementation handles real multimedia traffic, where clients stream content using the DASH protocol. To rigorously evaluate the performance of CAD in different settings, we implemented a configurable framework on top of Mininet [13]. Unlike network simulators, Mininet is a realistic network emulation environment built on Linux namespaces. Applications running on Mininet can be deployed on actual systems. Mininet has been used to evaluate research ideas at the network level e.g., [5], and application level e.g., [32].

Our Mininet framework emulates an ISP network of a telco-CDN, where the network topology is given as input to the framework. An ISP network is composed of points of presence (PoPs) at different geographical locations, where clients can access the Internet. We abstract the internal details of a PoP and implement it as a virtual switch using Open vSwitch 2.5.1 [33]. Links between virtual switches are created based on the input topology. The latency on a link between two switches is calculated based on the link medium and length. Switches and links process and carry real traffic using the Linux networking stack. An example ISP topology is shown in Figure 2, which is the AT&T topology in the East Coast of the US.[3] We use this topology in our experiments, because AT&T manages its own CDN [11]; hence, our setup is close to a real telco-CDN. We note that ISPs inter-connect with each other, typically at IXPs. Our framework emulates these inter-ISP links and measures the load on them, because this inter-ISP load represents an important cost factor for ISPs. For the example topology in Figure 2, the three PoPs at which AT&T inter-connects with other ISPs are marked by larger blue icons. The inter-connections occur through three IXPs: Telx Internet Exchange in Atlanta, Equinix Exchange in Chicago, and CoreSite Any2 Exchange in Boston.

The caching sites of the telco-CDN are located at PoPs. We implement each caching site as an HTTP server that can stream multimedia content using the DASH protocol. The characteristics of each caching site (e.g., storage and processing capacities) are provided as input to our framework. Caching sites receive and execute the distribution plans computed by CAD. The CAD algorithm itself is implemented and executed on a separate server inside the ISP. We implemented the streaming DASH clients using Go 1.7.4. The DASH clients request video representations from nearby caching sites according to the input demands. Finally, we manage the ISP network using OpenFlow [30]. In particular, we implemented an SDN controller in Python using Ryu 4.9. The SDN controller installs the TE rules computed by CAD into the switches.

[3] http://www.topology-zoo.org/dataset.html

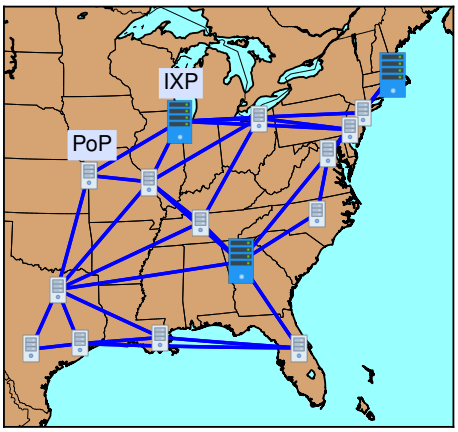


Fig. 2: The ISP topology used in our experiments. It is the AT&T network in the East Coast of the US.

B. Experimental Setup

We compare CAD against the algorithm in [15]. This algorithm reactively serves users using shortest path inside the network. We compare against this algorithm because it was shown in [15] that it outperforms other approaches. We refer to this algorithm as reactive CDN (rCDN). Both CAD and rCDN algorithms are used to manage the telco-CDN of the ISP topology in Figure 2, which has 16 locations.

We consider three important performance metrics that measure the network load and user QoE: (i) Amount of Inter-domain Traffic, which is the total bytes fetched from the origin server through inter-ISP links. (ii) Maximum Link Utilization (MLU), which is the 95-percentile of the traffic per second passing through an intra-domain link divided by its capacity. We use the 95-percentile because it is a good representation of the overall link performance [34]. (iii) Buffer Occupancy, which is the average number of video segments in the playback buffer at clients. High buffer occupancy values indicate better QoE for two reasons. First, because CAD does not change the client adaptation algorithm, high buffer occupancy may indirectly allow the client adaptation algorithm to request better quality representations [16]. Second, high buffer occupancy reduces the chance of rebuffering at clients.

There are $1,000$ videos each with up to 7 representations in the telco-CDN with a total size of 6 TB (i.e., the telco-CDN manages $7,000$ representations). These representations offer different quality levels, and they have average bitrates of 5, 3.5, 1.5, 0.5, 0.3, 0.2 and 0.1 Mbps. We transcode the master representation (5 Mbps) to create other representations using FFmpeg, and we measure the average processing time and file size for each representation. We limit FFmpeg processing to one CPU thread, and run each transcoding multiple times and take the average. According to DASH, each representation is divided into equal-size segments, 1 second each. We set the maximum allowed processing latency to 5 seconds.

We emulate a realistic behavior of users based on the analysis in [35]. Specifically, we consider four characteristics when generating user requests: (1) peak weekly demands, (2)

TABLE I: Arrival rates of requests for different periods.

	λ_1 (12-12:59am)	λ_2 (1am-3:59pm)	λ_3 (4-23:59pm)
Fri.	20	10	40
Sat.	35	30	70
Sun.	50	30	60

peak daily demands, (3) user behavior of quitting streaming sessions, and (4) representation popularity distribution. We generate user requests for three days: Friday, Saturday and Sunday, because they represent the peak weekly demands [35]. The request rate is not constant during the day, but it increases until its peak value in the evening. We employ the mixture probabilistic model that was observed in [35], which consists of three components representing early-quitters, drop-out users and steady viewers. Finally, the representation popularity is assumed to follow a Zipf-Mandelbrot distribution [36], which is a generalized form of the Zipf distribution. Unlike the Zipf distribution which is characterized by a single skew parameter α, the Zipf-Mandelbrot distribution has two parameters: skew parameter α and *plateau* parameter q, which captures the fact that multiple objects can have close popularity, which is more realistic. Large q values indicate that the total requests are spread on many videos, while small q values mean that most of the requests are concentrated in few, very popular, videos.

Putting all pieces together, we first generate user requests for each day using three Poisson processes with arrival rates λ_1 during early hours of the day, λ_2 during mid-day hours, and λ_3 during evening hours. Table I shows the λ values used in our experiments per second per region. When a user arrives, we randomly pick a representation following the Zipf-Mandelbrot distribution, and the number of segments the user will watch during the session using the mixture probabilistic distribution discussed above.

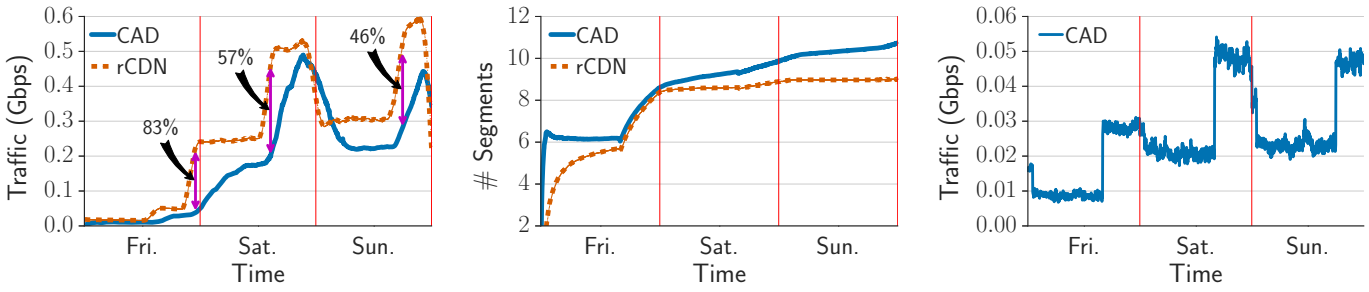
The key parameters we control and vary in our experiments are the storage capacity at each caching site and the popularity of video representations. The storage capacity ranges from 1% to 10% of the video library size. For the representation popularity, we fix the skew parameter of the Zipf-Mandelbrot distribution to $\alpha = 1.5$, but we change the plateau parameter q from 1 to 10. All other parameters in our setup are fixed as follows. Each caching site has 16 CPU cores with clock speed of 2.6 GHz. The capacity of inter-domain links is 400 Mbps and the average delay on them is 50 msec. The capacity of intra-domain links is 500 Mbps, and the delay is proportional to site distances. We repeat each experiment five times with different random seeds, and we report the average of the performance metrics across them.

The telco-CDN goes through two phases every day during our experiments. The first one is the *warm-up* phase, where caches are populated with videos either using CAD or rCDN. The second phase is the *streaming* phase, where clients request video segments and caching sites respond with the corresponding content according to the pre-computed distribution plans. We report the performance metrics during the streaming phase.

C. Results

CAD Outperforms the Closet Algorithm in Literature: We start by comparing the performance of CAD versus rCDN under different storage capacities per caching site. We set the plateau parameter of the video representation popularity distribution $q = 5$. A representative sample of our results is depicted in Figure 3, where we show how the amount of inter-domain traffic and buffer occupancy change throughout the 3-day period of the experiment. The results in Figure 3 are for $q = 5$ and storage capacity of 10% of the video library size. Similar results were obtained for other values of the storage capacity and plateau parameter q. Figure 3a depicts the time series of the inter-domain traffic. Vertical arrows show the maximum improvements achieved by CAD for every day. On Friday, both CAD and rCDN result in low inter-domain traffic, because most demands can be handled inside the network. CAD, however, results in much lower inter-domain traffic compared to rCDN. On Saturday, the inter-domain traffic increases for both rCDN and CAD till early hours of the evening, because of the increase in the demands. However, CAD outperforms rCDN in terms of inter-domain traffic and buffer occupancy as shown in Figures 3a–3b. In the evening, rCDN results in more requests to the origin server and increases the end-to-end latency. As a result, CAD improves the buffer occupancy compared to rCDN. The same results are observed on Sunday. Specifically, CAD reduces the inter-domain traffic by up to 83%, 57% and 46% on Friday, Saturday and Sunday, respectively. These improvements are instantaneous and may not reflect the overall performance. Thus, we calculate the resulting improvement of CAD per day in terms of total inter-domain traffic. In particular, CAD reduces the total inter-domain traffic by 64%, 32% and 23% on Friday, Saturday and Sunday, respectively. Both CAD and rCDN result in same MLU of 21%. This is because CAD jointly manages the network and caching resources while fetching or creating representations on-demand. In addition, CAD increases the number of segments at client buffers by up to 14%.

Impact of Active Distribution: To shed some lights on the importance of the *active distribution* feature of CAD, we plot in Figure 3c the traffic served by active distribution in CAD (i.e., creating representations on demand). This figure shows the difference in inter-domain traffic when CAD employs only cooperative distribution and when it uses both cooperative and active distribution. Specifically, without active distribution, the traffic shown in Figure 3c would have been requested from the origin server. In terms of the total inter-domain traffic, the on-demand processing contributes to about 32% of the improvement in inter-domain traffic. In particular, CAD reduces the total inter-domain traffic by 138 Gb compared to rCDN. In Figure 3c, the total traffic created on-demand is 45 Gb, which is 32% of the improvement in the inter-domain traffic. For daily inter-domain traffic, active distribution contributes to about 50%, 27% and 35% of the improvements on Friday, Saturday and Sunday, respectively.



(a) Inter-domain Traffic (b) Buffer Occupancy (c) Traffic Served by Active Distribution

Fig. 3: Time series of inter-domain traffic, buffer occupancy, and on-demand created traffic over the 3-day experiment period.

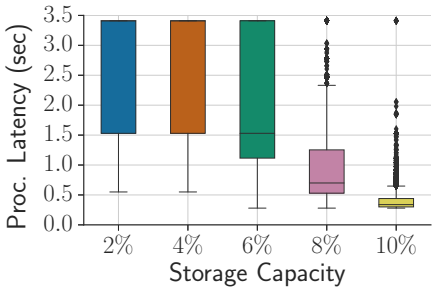


Fig. 4: Processing latency distributions of CAD to create representations on-demand under different storage capacities.

CAD Respects the Maximum Processing Latency: For every storage capacity setting, we plot the processing latency distribution of CAD to create video representations on-demand as a box plot in Figure 4. The box plot is a compact visualization of the three quartiles 25%, 50% and 75%, minimum and maximum values, and the outliers for a given distribution (i.e., the processing latency in our case). For example, when the storage capacity is 8% of the video library size, 25%, 50% and 75% of the created representations on-demand are processed in less than 0.5, 0.65 and 1.25 seconds, respectively. Moreover, the minimum and maximum processing latencies are 0.25 and 2.4 seconds. The figure shows that CAD *strictly* respects the maximum allowed processing latency set by the operator (i.e., 5 seconds). It also shows that, as the storage capacity increases, CAD stores and fetches more representations as indicated by the reduction in the latency. This ensures that the end-to-end latency is minimized as the telco-CDN resources allow.

Impact of Varying Popularity Patterns: We assess the impact of various popularity patterns of the video representations. We conduct three experiments where we set the plateau parameter of the Zipf-Mandelbrot distribution q to: 1 (highly skewed popularity), 5 (moderate skewness), and 10 (less skewness, i.e., more objects have similar popularity). We set the storage capacity to 10% of the video library size.

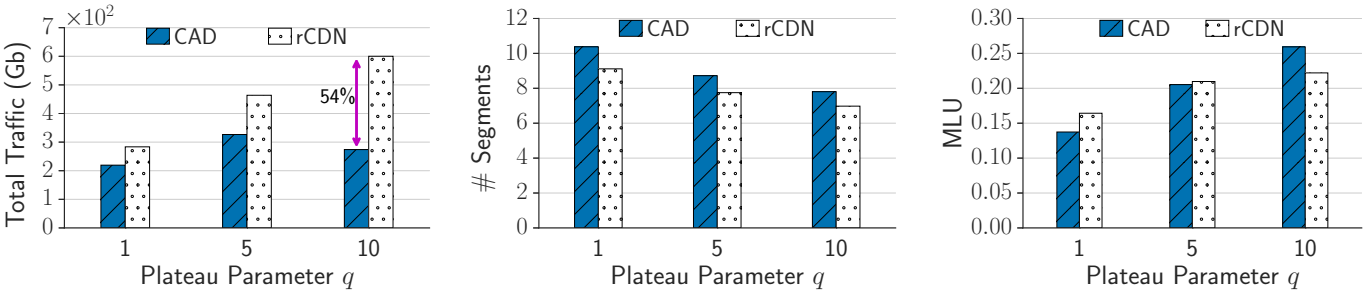
Figure 5 depicts the averages of the three considered performance metrics across the entire three-day period. The figure shows that CAD consistently outperforms rCDN in terms of amount of inter-domain traffic and buffer occupancy. Figure 5a shows the inter-domain traffic results. CAD outperforms rCDN by up to 54% when $q = 10$. This is because decreasing skewness (i.e., increasing q) gives CAD the room to fetch and create more representations on-demand inside the telco-CDN. In term of the buffer occupancy at clients, CAD outperforms rCDN by up to 14% when $q = 1$. As q increases (i.e., less skewness), the buffer occupancy decreases because more requests are served by the origin server as Figure 5a shows. Finally, the MLU of CAD increases from 22% to 26% when $q = 10$ compared to rCDN as depicted in Figure 5c. This is because decreasing skewness allows CAD to fetch and create more representations on-demand. This small increase in MLU does not diminish the gain of reducing the utilization of the more expensive inter-domain links by 54% [8].

Running Time of CAD: We measured the running time of CAD, which varied from 46 seconds to 140 seconds to calculate the results of a whole day. The running time varies based on the expected demands for each day. This shows two important aspects. First, CAD is practical and can run on commodity servers. Second, it gives the telco-CDN operator room for reacting upon demand variations and network failures. For example, the operator can reduce the length of CAD period from 24 hours (as in our experiments) to order of tens of minutes.

Summary: CAD outperforms the closest work in terms of inter-domain traffic by a wide margin (up to 64%). In addition, CAD has modular functions and improvements. That is, CAD can be deployed with no on-demand transcoding. As shown in Figure 3c, the on-demand creation results in 32% of the improvements in the total inter-domain traffic, the other 68% improvements are due to the careful TE and content distribution done by CAD. That is, if caching sites have processing resources, CAD uses them to improve the inter-domain traffic. Otherwise, CAD will rely on careful TE of network paths.

VI. Conclusions

We considered the problem of managing the resources of the emerging telco-CDNs, which are content distribution networks operated by ISPs. We proposed a new algorithm called CAD (short for Cooperative Active Distribution) to solve this problem. The key new ideas in CAD include: (i)



(a) Inter-domain Traffic (b) Buffer Occupancy (c) Maximum Link Utilization (MLU)

Fig. 5: Average metrics of CAD and rCDN under different popularity patterns. Storage capacity is 10% of the library size.

jointly optimizing the utilization of the caching resources and the selection of traffic paths within the network, (ii) creating video representations on-demand, and (iii) enabling cooperation among caching sites to serve as much as possible of the multimedia traffic within the ISP. We implemented and evaluated CAD in a Mininet-based emulation network, which carries real traffic using SDN-managed virtual switches. We also implemented caching servers and DASH clients, and used them to serve and request video segments. Our results show that compared to the closest work in the literature, CAD achieves up to 64% reduction in the inter-domain traffic, which is a major cost factor for ISPs. CAD also improves the number of segments in the client buffer by up to 14%, which is an important metric to measure user QoE. Finally, CAD chooses the traffic paths carefully to avoid introducing bottlenecks in the ISP network. These improvements are critical for ISPs and content providers who collaborate, especially with the growth of the viewership of these content providers.

Acknowledgments

This work is supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and by the Qatar National Research Fund (grant [NPRP8-519-1-108]).

References

[1] "The Zettabyte Era: Trends and Analysis," https://bit.ly/295hnsl, [Online; accessed Jan 2019].
[2] E. Nygren *et al.*, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, 2010.
[3] B. Frank *et al.*, "Collaboration Opportunities for Content Delivery and Network Infrastructures," *ACM SIGCOMM eBook on Recent Advances in Networking*, vol. 1, 2013.
[4] N. Spring *et al.*, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 1, 2004.
[5] M. Malboubi *et al.*, "Intelligent sdn based traffic (de)aggregation and measurement paradigm (istamp)," in *Proc. of IEEE INFOCOM'14*.
[6] M. A. Sanchez *et al.*, "Inter-domain traffic estimation for the outsider," in *Proc. of ACM IMC'14*.
[7] Y. Liu *et al.*, "On the interaction between overlay routing and underlay routing," in *Proc. of IEEE INFOCOM'05*.
[8] H. Nguyen *et al.*, "Network loss inference with second order statistics of end-to-end flows," in *Proc. of ACM IMC'07*.
[9] B. Frank *et al.*, "Pushing cdn-isp collaboration to the limit," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, 2013.
[10] "Netflix OpenConnect," https://openconnect.netflix.com/en/, Netflix, April 2017, [Online; accessed Jan 2019].
[11] "AT&T-Content Delivery Network Services." http://soc.att.com/1T4sXmB, AT&T, April 2017, [Online; accessed Jan 2019].
[12] "Cisco Report on CDN Federation," http://bit.ly/2oTT8bp, Cisco, October 2012, [Online; accessed Jan 2019].
[13] N. Handigol *et al.*, "Reproducible network experiments using container-based emulation," in *Proc. of ACM CoNEXT'12*.
[14] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles," in *Proc. of ACM MMSys'11*.
[15] A. Sharma *et al.*, "Distributing content simplifies isp traffic engineering," in *Proc. of ACM SIGMETRICS'13*.
[16] T.-Y. Huang *et al.*, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. of ACM SIGCOMM'14*.
[17] R. Teixeira *et al.*, "Dynamics of hot-potato routing in ip networks," in *Proc. of ACM SIGMETRICS'04*.
[18] "Streaming Video Alliance announces first trials of the alliance's compliant open caching systems deployed in ISP networks," http://bit.ly/2pz5zqm, Streaming Video Alliance, January 2017, [Online; accessed Jan 2019].
[19] H. Xie *et al.*, "Tecc: Towards collaborative in-network caching guided by traffic engineering," in *Proc. of IEEE INFOCOM'12*.
[20] W. Jiang *et al.*, "Cooperative content distribution and traffic engineering in an isp network," in *Proc. of ACM SIGMETRICS'09*.
[21] M. Ammar *et al.*, "A vision for zero-hop networking (zen)," in *Proc. of IEEE ICDCS'17*.
[22] Z. Li *et al.*, "In a telco-cdn, pushing content makes sense," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, 2013.
[23] F. Zhou *et al.*, "Joint optimization for the delivery of multiple video channels in telco-cdns," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, 2015.
[24] G. Gao *et al.*, "Resource provisioning and profit maximization for transcoding in clouds: A two-timescale approach," *IEEE Transactions on Multimedia*, vol. 19, no. 4, 2017.
[25] D. K. Krishnappa *et al.*, "Optimizing the video transcoding workflow in content delivery networks," in *Proc. of ACM MMSys'15*.
[26] H. Ma *et al.*, "Dynamic scheduling on video transcoding for mpeg dash in the cloud environment," in *Proc. of ACM MMSys'14*.
[27] A. Ganjam *et al.*, "C3: Internet-scale control plane for video quality optimization," in *Proc. of NSDI'15*.
[28] D. Crankshaw *et al.*, "Clipper: A low-latency online prediction serving system," in *Proc. of USENIX NSDI'17*.
[29] A. Goldberg *et al.*, "Solving minimum-cost flow problems by successive approximation," in *Proc. of ACM STOC'87*.
[30] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, 2008.
[31] C. Filsfils *et al.*, "Segment Routing Architecture," RFC 8402, July 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8402.txt
[32] A. Bentaleb *et al.*, "Sdndash: Improving qoe of http adaptive streaming using software defined networking," in *Proc. of ACM MM'16*.
[33] B. Pfaff *et al.*, "The design and implementation of open vswitch," in *Proc. of USENIX NSDI'15*.
[34] C. Huang *et al.*, "Can internet video-on-demand be profitable?" in *Proc. of ACM SIGCOMM'07*.
[35] A. Balachandran *et al.*, "Analyzing the potential benefits of cdn augmentation strategies for internet video workloads," in *Proc. ACM IMC'13*.
[36] M. Hefeeda *et al.*, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, 2008.