Scalable Request Routing for VR-ready CDNs

Pierre-Louis Poirion
and Jérémie Leguay
Mathematical and Algorithmic Sciences Lab
France Research Center, Huawei Technologies Co. Ltd.
Email: firstname.name@huawei.com

Liu Ruosi
Network Technology Lab
Huawei Technologies Co. Ltd., Schenzen, China
Email:liuruosi@huawei.com

***Abstract*—Content Delivery Networks (CDN) are witnessing the outburst of video traffic and must get ready for streaming services based on Virtual Reality (VR). The immersive experience that VR provides for live events or video on demand dramatically increases the need for network and computing resources and VR sessions are foreseen to be elephants flows in service providers' infrastructures. This paper analyzes the challenges of request routing in VR-ready CDNs and argues that decisions must be taken at a finer granularity compared to today's practice. To make such decisions at scale, we propose a hybrid control framework, called VR-ready Admission Control (VRAC), to decompose and solve the underlying optimization problem in real-time. We demonstrate in a realistic setting that it will be decisive to withstand the ever increasing penetration of data-hungry VR video streams.**

I. Introduction

Globally, IP video traffic is expected to represent 82 percent of all IP traffic by 2020 [1]. While a large variety of Video-on-Demand (VoD) and video-streaming services have emerged in the past years, the uses continue to evolve rapidly. Virtual Reality (VR) has the potential to become the next revolution to video, similar to what color has been to black and white. VR-based videos are captured with a 360° stereoscopic camera and viewed through a specific headset where users can see and hear the scene all around them.

Content Delivery Networks (CDN) have prevailed as the dominant method to deliver content. 72 percent of Internet video traffic is intended to cross CDNs by 2019 [1]. The largest over-the-top player Akamai currently has over 170,000 edge servers located in over 1300 networks in 102 countries [2]. And large CDNs are already serving billions of content requests per day. However, the emergence of VR-based video streaming services is expected to dramatically increase the need for networking and computing resources. CDNs solutions will have to evolve to cope with the rise of such real-time and data hungry video services. In fact, in the next years, VR sessions are foreseen to reach throughputs in the order of Gbps and to become elephant flows in service providers' infrastructures. Even if the penetration rate of VR streaming services is still low, Goldman & Sachs [3] foresees that by 2020, 28 and 24 millions users will respectively consume live broadcasting and video entertainment services.

Current CDN solutions are composed by a network of caches storing the most popular contents and a user mapping system that redirects client requests to the best proximal cache server [4]. The user mapping [2], [5] is typically implemented on top of the Domain Name System (DNS) as it eases content integration in web applications and offers reliability and performance at Internet scale. Such system enforces request routing decisions at the granularity of clients' local DNS or IP prefix [5]. However, VR streams are expected to consume a significant amount of bandwidth and processing, their admission into a CDN infrastructure must be carefully decided to be consistent with available resources. Making aggregated decisions will not work and lead to the rejection of most demands.

Our paper analyzes the challenges of request routing in VR-ready CDNs and shows that the problem becomes an online packing problem where custom decisions must be taken for every request. To solve the problem at scale and in real-time with tens of millions of requests per day, we propose a hybrid control framework, called VR-ready Admission Control (VRAC), which uses advanced clustering and online optimization techniques. Our approach extends the control framework introduced in [6] for the maintenance of distribution trees in video CDNs. More specifically, we propose 1) to break the large admission control problem into several smaller ones by assigning clients to clusters of CDN servers, and 2) to adopt a two step decision process for the routing of VR requests. Streaming requests are first directed to the best proximal cluster using DNS mappings. In a second step, they are processed in batch by cluster heads to optimize their allocation in an online fashion using a fast linear programming based heuristic. We show on a realistic scenario that VRAC outperforms current solutions, especially when the system is highly loaded and do not have the capacity to accept all requests.

II. Challenges

This section analyses the characteristics of VR streams and the request routing problem in VR-ready CDNs.

A. *Characteristics of VR streams*

The vision of users in the virtual environment evolves in a sphere, which unfolds 360° horizontally and 180° vertically. Users only view a part of the spherical data called the Field of View (FoV). Fig. 1 shows that on average, humans are able to see about 220° horizontally of FoV naturally, while modern consumer-grade head-mounted displays (HMDs) have a FoV

	Entry-Level VR	**Advanced-level VR**	**Ultimate-level VR**
Continuous experience duration	Less than 20 minutes	20 to 60 minutes	Over 60 minutes
Estimated time	Now-2 years	3-5 years	5-10 years
Video resolution	Full view 8K 2D	Full view 12K 2D	Full view 24K 3D
Full frame resolution	7680*3840	11520*5760	23040*11520
Single-lens resolution	2K	4K	8K
Frame rate	30/s	60/s	120/s
Video bitrate (360° scheme)	64 Mbps	279 Mbps	3.29 Gbps
Video bitrate (FoV scheme)	12.8 Mbps	55.8 Mbps	568 Mbps
Requirement on access bandwidth	100 Mbps	1 Gbps	2-5 Gbps

TABLE I
CHARACTERISTICS OF VR-BASED VIDEO STREAMS.

within 90° and 120° horizontally and vertically [7]. The figure also presents what is displayed on the screen compared to what is perceived by the human brain.

VR videos are captured by mounting several cameras with a high degree of FoV overlap on dedicated rigs [1] (between 6 to 14 cameras, typically). For a 120° FoV resolution of 4K, the 360° video can be up to 24K which induces bit rates in the order of 1 Gbps. The video can be encoded as a single stream. However, as such streams consume a tremendous amount of bandwidth, only a few Internet users can afford to watch them at the moment. For these reasons, specific projection techniques such as *pyramid encoding* [2] are under development to offer a low degree of distortion and a low consumption of resources. Using the scheme proposed by Facebook, a 360° video is transformed into the pyramid format for 30 viewing regions and 5 different resolutions, making a total of 150 different sub-streams. While this scheme requires more storage, it helps reducing network resource consumption up to 80%. The downside is that it requires a tight coordination between the headset and the streaming server to switch between sub-streams. For a seamless user experience, it is commonly accepted that the motion-to-photons latency, i.e. the delay between head movements and video updates on the screen, cannot exceed 20 ms [8].

To offer the best user experience under constrained network and computation resources [9], two classes of transmission schemes are currently envisioned:

[1]https://www.360rize.com/

[2]https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/

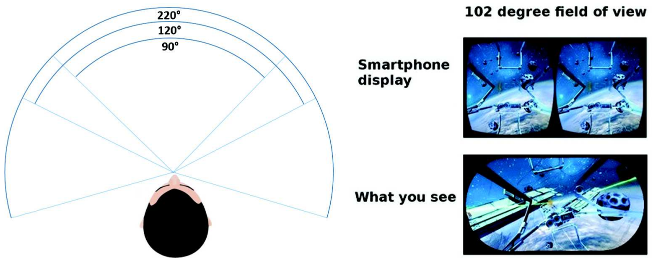

Fig. 1. Field of Fiew (FoV) using Head Mounted Displays (HMD)

- 360° scheme: the full 360° video is transmitted to the terminal which in turn extracts at a low complexity the right visual data to display.
- *FoV* scheme: only the visible part of the video is transmitted to decrease bandwidth. The 360° video is split into several sub-streams which are downloaded according to user position. As the terminal needs to switch between sub-streams, the maximum network latency is a few milliseconds.

Table I presents in details the foreseen characteristics of VR traffic [9] at three time horizons: *Entry-level* (today), *Advanced-level* (3 years from now) and *Ultimate-level* (5 years from now). Videos available today (*Entry-level*) are encoded at a bitrate of 64 Mbps for a full 360° view, while *Ultimate-level* VR videos are expected to be in the order of a few Gbps. Considering state of the art encoding techniques such as pyramid encoding, we consider a 80% reduction of the video bitrate for *FoV* compared to 360°. Although these figures might decrease as new encoding or streaming features are released, they are in-line with current estimations from VR stakeholders. In the same time period (next five years), Cisco's VNI [1] predicts that broadband Internet access for fixed networks will grow at a much slower pace. For instance, the average access bandwidth is expected to increase in North America from 37.6 Mb/s in 2017 to 51.4 Mb/s in 2020. This confirms that VR streams are very likely to be considered as elephant flows in operators and CDNs infrastructures.

Flash crowd events are expected to amplify this phenomena and generate remarkable tidal effects. Indeed, for live events the audience may raise suddenly and lead to a 100 times increase of the difference between peak traffic and background traffic [9].

B. *Impact for request routing*

Request routing is the process of directing client requests to the best CDN servers [10]. In general, it aims at avoiding the servers that provide low performance (e.g., poor processing and downloading time) while trying not to overload the others.

Routing decisions are typically updated by the CDN operator every few minutes solving a stable marriage problem with tree constraints (see Akamai's paper [2]). These decisions are enforced through DNS mappings between clients and

CDN servers. And they are defined at the granularity clients' local DNS or IP prefix. The stable marriage algorithm takes input parameters such as the average network capacity and latency, which are both estimated from past requests. It also considers pre-configured routing preferences and the status of the infrastructure (e.g., current server load, capacity). Then, when a client requests a specific content, it resolves the content URL and retrieves from the DNS system a candidate list of typically 2 or 3 CDN servers. If it fails to download the content from the first server, because of a cache miss for instance, the client tries from the second one, and so on.

As explained in Sec. II-A, VR sessions are expected to consume a significant amount of bandwidth and processing. As a consequence, their admission must be carefully decided. Making decisions at a coarse granularity, like in traditional CDNs, will not work and lead to the rejection of most demands. Ideally, a centralized entity should handle elephant requests and properly pack them into the infrastructure. However, the optimization problem is generally too large for a resolution in real-time. In practice, the challenge is to find a solution that scales.

The elephant nature of VR flows also calls for a tighter coordination between ISP and CDN providers so that network resources can be dynamically allocated between servers and clients. As explained in [9], this requires a significant evolution of access and transport networks so that end to end bearers can be setup in a few milliseconds. In our paper, we consider the current situation where there is no dynamic cooperation.

C. *Problem formulation*

In this section, we formulate the request routing optimization problem in order to maximize the number of VR sessions served by the CDN.

Let $V = \{V_1, ..., V_n\}$ and $Q = \{Q_1, ..., Q_k\}$ be respectively the set of servers and requests. $E = \{(i, j) \mid i \in V, j \in Q\}$ denotes the set of possible request-server routing and $G = (V, Q, E)$ the corresponding bipartite graph. Like in current CDNs, the existence of a link (i, j) in G is determined by routing preferences which depend on the relative geographical position of clients and servers or on peering agreements. For all existing links (i, j) we denote by l_{ij} their round-trip latency. The maximum bandwidth that each server i can output is denoted by b_i and each server cannot handle more than π_i requests in parallel.

For each request j, we denote by r_j its bandwidth. The transmission scheme used by each request is decided a priori: for each possible server i, the system checks if the round-trip latency allows the use of *FoV* (i.e., if $l_{ij} \leq 10$ ms). If this holds true for at least one server, we select *FoV* as it consumes the lowest amount of resources and our objective is to maximize the number of accepted requests.

We can now model the request routing problem as an optimization one. For all $(i, j) \in E$ we denote by x_{ij} the binary variable equal to one if and only if request j is redirected on server i. The problem can hence be formulated as the following packing problem $\mathcal{P}$:

$$(P)\begin{cases} \max\limits_{x} & \sum\limits_{(i,j)\in E} x_{ij} \\ & \sum\limits_{j\in Q} r_j.x_{ij} \leq b_j \quad \forall i \in V & (1) \\ & \sum\limits_{j\in Q} x_{ij} \leq \pi_j \quad \forall i \in V & (2) \\ & \sum\limits_{i\in V} x_{ij} \leq 1 \quad \forall j \in Q & (3) \\ & x_{ij} \in \{0,1\} \quad \forall i \in V, \forall j \in Q \end{cases}$$

Constraint (1) imposes the maximum bandwidth capacity of each server node; constraint (2) limits the number of parallel requests on a server; constraint (3) models the fact that each request can be assigned to at most one server. The objective function of the Integer Linear Program (ILP) above maximize the number of redirected requests.

Although the overall problem can be modeled by the Integer Linear Program (ILP) above, this formulation requires prior knowledge of all future requests, hence it represents an *offline* problem. In practice, the CDN discovers arrivals and departures of requests when they occur and sequential decisions must be taken to accept or reject requests. Moreover, the controller cannot revisit past decisions. The request routing problem is therefore the online version of this problem. In this particular setting, the goal of *online algorithms* [11] is to be as *competitive* as possible with regards to the offline optimal, impossible to attain in practice.

III. Scalable Request Routing

As mentioned in Sec. II-B, DNS has a number of advantages. Thus, we believe that it should remain a key component of the system. Indeed, it eases content integration in web applications and helps solving web-scale load balancing problems. For this reason, we propose VR Admission Control (VRAC), a two step request routing approach which extends current routing solutions based on DNS mappings.

As the rate of received requests by the CDN can be high, VRAC adopts a batch processing approach to optimize the request allocation, under the constraint that decisions must be taken in less than one second for each batch. Indeed, the goal is to accept a maximum number of requests while being responsive to clients. Responsiveness is an important requirement as users are waiting for their VR session to start. As depicted in Fig. 2, we also group CDN servers into clusters by partitioning the bipartite graph G with the objective of solving smaller optimization problems inside clusters. The outcome of the clustering procedure are DNS mappings which connect clients' IP prefixes to VRAC cluster heads. In this way, routing requests are directed towards the best proximal group of servers based on coarse grained and quasi static information. Then, in a second step, cluster heads determine precisely which CDN servers[3] must process requests.

As the optimization problem for each batch is too large for a global resolution to optimality in real-time, the cluster

[3]To simplify the presentation, we reduce to *a server*, a CDN node which may actually contain several physical servers.

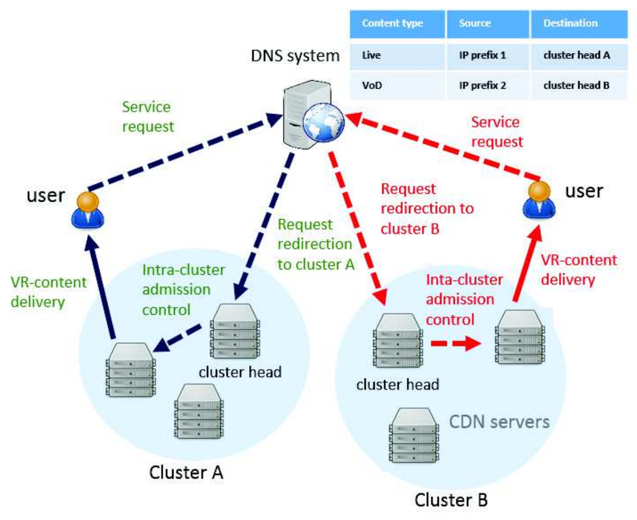

Fig. 2. Two step request routing in VRAC: DNS directs requests to the best cluster head which refines the decision.

approach helps to efficiently decompose the problem. In addition, as presented in Sec. III-B, an efficient heuristic is introduced to quickly obtain good allocations inside a cluster and take the online aspect of the problem into account.

A. *User mapping to clusters*

To obtain a tractable problem we periodically decompose the bipartite graph G into several clusters. Hence, we break the large request admission problem into several smaller ones, that could be solved in parallel by cluster heads. Since routing requests come from different network regions, it makes sense to assume that different sets of users can be linked to disjoints sets of servers.

A natural idea to achieve this decomposition is to find a minimum cut C of G (see Fig. 3). More formally, we seek for a subset $C \subset V \cup Q$ such that

$$cut(C) = \sum_{p \in C,\ q \notin C} A_{pq}$$

is minimized (where A is the adjacency matrix of the graph G). Such a minimum cut C^* would induce a first partition $(C^*, \bar{C}^*)$ of the graph where the number of connections between C^* and $\bar{C}^*$ is minimized. We then use this approach recursively, to cluster the graph is several subgraphs. Notice that the minimum cut problem is a polynomial problem that can be solved efficiently with the Stoer and Wagner algorithm: [12].

The output of the clustering defines DNS mappings. As it operates at a coarse granularity, they can be updated at a slow pace (e.g., a few times a day).

B. *Intra-cluster admission*

In this section we explain how to solve the routing problem inside a cluster of n servers.

We now assume that we are in an *online* setting: at each time step t, a new batch of k_t incoming requests has to be

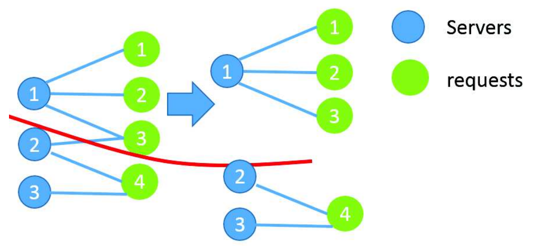

Fig. 3. A minimum cut that clusters the bipartite graph in two.

routed to the n servers, such that the constraints described in Sec. II-C are satisfied. The goal is now to maximize the number of accepted requests in the whole time windows $[0, T]$ considered.

One of the strengths of our approach is that we do not aim to maximize the number of accepted requests at each time step $t \in [1, ..., T]$, as it would overload very quickly the system by *pack* as many requests as possible in the clusters at the beginning of the process. Instead of being greedy, we adapt and generalize the approach proposed in [13] to solve an online packing optimization problem which consists in *saving resources* for upcoming requests.

Given a vector $\alpha \in \mathbb{R}^n$ of coefficients belonging to $[0, 1]$, we define the optimization problem $\mathcal{P}(\alpha)$ as the assignment problem, defined in Sec. II-C, where the capacity of server i has been scaled by $0 \leq \alpha_i \leq 1$, i.e., b_i is replaced by $\alpha_i.b_i$ in equation (1).

Furthermore, to solve the integer linear program $\mathcal{P}(\alpha)$ at each time period in a short amount of time (less than one second in practice), we first solve its linear relaxation $\tilde{\mathcal{P}}(\alpha)$, where the binary variables $x_{ij} \in \{0, 1\}$ have been relaxed to $0 \leq x_{ij} \leq 1$. From an optimal solution $\tilde{x}$ of $\tilde{\mathcal{P}}(\alpha)$ we obtain a first assignment, $\bar{x}_{ij}$, by randomly rounding $\tilde{x}_{ij}$: for each request j, we assign it to server i with probability $\tilde{x}_{ij}$, i.e., $\Pr(\bar{x}_{ij} = 1) = \tilde{x}_{ij}$. Since $\bar{x}_{ij}$ might be infeasible (with respect to the capacity constraints), we then modify the server-request assignment imposed by $\bar{x}_{ij}$ by solving a very small knapsack problem. Our algorithm can be summarized below:

We choose the parameter α such that we can save some space for future requests. More precisely:

- **Transition stage:** Until no request that has been previously assigned to a server ends, we keep saving space on the servers by dividing equally the bandwidth on the servers by the estimated length of this period.
- **Stationary stage:** when assigned requests start to leave, we estimate, on each server, at each time step, the average bandwidth that is leaving the server (*Leaving load*) and the average bandwidth that is assigned to the server (*Entering load*). We use these two values to design α and restraint the network to assign requests on servers that are already almost full.

Notice that α is a *learning* parameter that learns, over time, how much space should be saved for later on each server.

Algorithm 1: Assignment algorithm in VRAC

```
  Output: Assign requests to servers on $[0, T]$
  /* For each time step                              */
1 for $t \in [1, ..., T]$ do
      /* Solve the relaxed problem                   */
2     Solve $\tilde{\mathcal{P}}(\alpha)$; get $\tilde{x}_{ij}$
      /* Use random rounding and get a
         first (possibly) infeasible
         assignment                                  */
3     $\bar{x}_{ij} \leftarrow \forall j \in [1, ..., k_t]$ assign $j$ to $i$ with probability
      $\tilde{x}_{ij}$
      /* Deduce a feasible assignment                */
4     For each server, solve a knapsack to satisfy as many
      assignments as possible in $\bar{x}_{ij}$
      /* Update the coefficients $\alpha$            */
5     $\forall$ server $i$, $\alpha_i = \frac{\text{Leaving load}}{\text{Entering load}}$
```

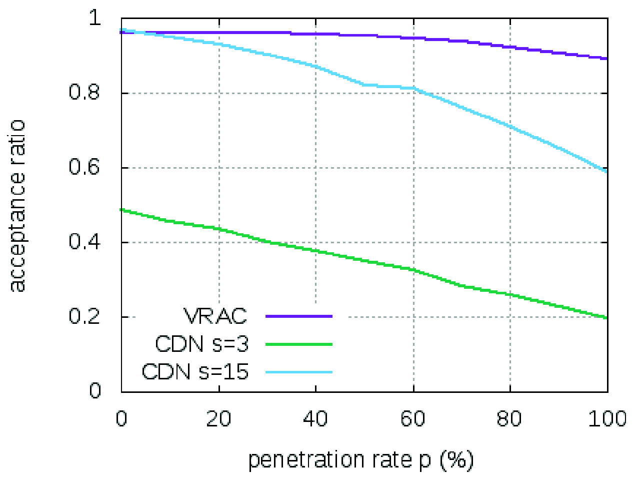

Fig. 4. Average acceptance ratio function of the penetration rate of Advanced-level users.

IV. Performance Results

This section details our experimental setup and the results we have obtained.

A. Simulation setup

For the experiments, we consider a VR-ready CDN composed by 10 clusters. Each cluster contains 100 servers for approximately 10^6 users. We assume that the clustering of CDN servers is already decided and we simulate the admission control algorithm in one cluster. We estimate that at each server, 100 requests arrive every minute (the maximum number of requests per server is never attained). We model the arrival instants of requests with a Poisson process such that in average 166 requests arrive per second per cluster (i.e., 100 requests / minute / server or 144 million requests per day for the whole CDN composed of 1000 servers). Furthermore, we estimate that clients can only be served by 30% of servers. In practice, the possibility to connect clients and servers is defined by quasi-static preferences as for the computation of DNS mappings.

The round-trip latency between clients and servers follows a normal distribution with an average and a variance of both 20 ms. In our simulations, about 40% of users are served with *FOV* as the maximum round trip latency for this scheme is 10 ms. We take the average video bitrates from Table I and consider that due to variable bitrates encoding, the size of requests follows a normal distribution where the standard deviation equals to 10% of the average. Similarly to [9], we assume that the network bitrate is 1.5 times higher than the video bitrate. Finally, the duration of requests is exponentially distributed with an average of 5 minutes.

We compare our solution to *CDN*, a request routing scheme similar to what is implemented today [2]. In this scheme, DNS mappings are updated every minute solving a stable marriage problem. Each request receives a list of s servers from which the content can be retrieved. The client initiates a VR session with the first server and decides to switch from 360° to *FoV* if the latency is below 10 ms. In case the session cannot be established, the client tries with the second one, and so on. We considered the current situation where $s = 3$ and a possible future adaptation of the scheme with $s = 15$ that tries to accept more requests when the system is highly loaded.

We simulated 1000 seconds so that the system arrives in steady state. To obtain performance forecast of VR-ready CDNs, we increase the penetration rate of VR technologies by we varying the percentage p of requests using *Advanced-level* VR. The rest of the requests (1-p) are at *Entry-level*.

We implemented the simulator using Julia and used CPLEX to solve the relaxed problem and the small sub-knapsack problems. All experiments were performed on a 2.7 GHz Intel i7 dual-core CPU with 16 GB RAM. Simulation results are average results over 5 runs.

B. Numerical results

Fig. 4 plots the average ratio of accepted requests over the whole simulation as function of p. VRAC clearly outperforms *CDN* and accepts almost all requests, although its acceptance ratio falls to 90% when all users are at *Advanced-level*. We believe that the evolution of Internet broadband access will compensate the inflation of VR traffic so that almost 100% of requests can be accepted with a scheme like VRAC. Looking at the fast degradation of the performance of current solutions ($s = 3$) for request routing, we see that VR requests must be processed individually with efficient online packing algorithms such as the one we proposed. The adaptations of *CDN* with $s = 15$ improves the performance but the gap with VRAC increases significantly with p. In addition, such adaptations require a lot of trial and error accesses to CDN servers and may induce a slow responsiveness.

Fig. 5 shows the acceptance ratio over time for a penetration rate of 80%. We also show the average remaining bandwidth capacity (for VRAC) of the servers over time. We see that the

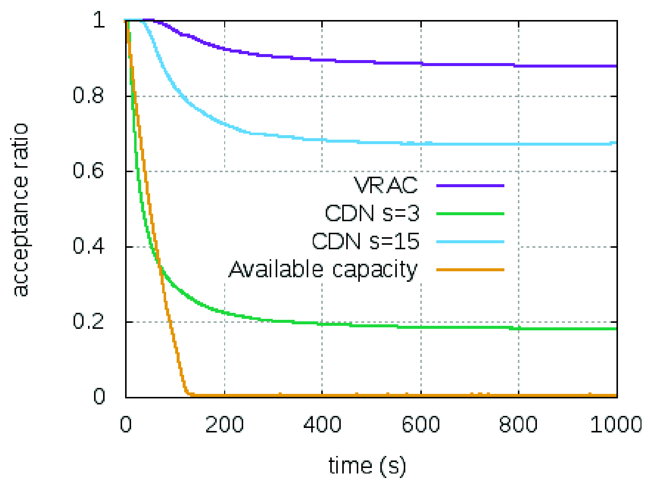

Fig. 5. Acceptance ratio over time with a penetration rate of 80%.

steady state is reached at approximatively $t = 200$ s. In term of execution times in all simulations, we saw that it takes in average 412 ms to process one batch of requests with VRAC. The maximum amount of time encountered is 847 ms which meets our requirement of one second.

V. Conclusion

In this paper we proposed a hybrid control framework, called VR-ready Admission Control (VRAC), to decompose and solve the request routing problem at scale for VR-ready CDNs. We demonstrated in a realistic setting that our method outperforms current algorithms when the system is highly overloaded. Our results show that taking custom decisions for every request is decisive to withstand the ever increasing penetration rate of data-hungry VR video streams. In particular, we saw that when the percentage of users using advanced-VR is increasing the classical DNS-based solutios based on a stable marriage algorithm was not able to route a sufficient number of requests, while our algorithm could maintain a success ratio above 80%.

Future works along these lines include the integration of advanced traffic predictions in the computation of the *learning* parameter. We also aim to extend the problem with the dynamic allocation of network resources between clients and servers.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2014 - 2019," May 2015.

[2] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *ACM SIGCOMM CCR*, vol. 45, no. 3, pp. 52–66, 2015.

[3] "Virtual and Augmented Reality: Goldman Sachs Report (2016)."

[4] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[5] F. Chen, R. K. Sitaraman, and M. Torres, "End-user mapping: Next generation request routing for content delivery," in *ACM SIGCOMM CCR*, vol. 45, no. 4, 2015, pp. 167–181.

[6] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang, "Practical, real-time centralized control for cdn-based live video delivery," *ACM SIGCOMM CCR*, vol. 45, no. 4, pp. 311–324, 2015.

[7] P. Savino and H. Danesh-Meyer, *Color Atlas and Synopsis of Clinical Ophthalmology–Wills Eye Institute–Neuro-Ophthalmology*. Lippincott Williams & Wilkins, 2012.

[8] M. Abrash, "What vr could, should, and almost certainly will be within two years," *Steam Dev Days, Seattle*, 2014.

[9] "VR-Oriented Bearer Network Requirements: Huawei's report (2016)."

[10] F. Thouin and M. Coates, "Video-on-demand networks: design approaches and future challenges," *IEEE network*, vol. 21, no. 2, 2007.

[11] N. Buchbinder, J. S. Naor *et al.*, "The design of competitive online algorithms via a primal–dual approach," *Foundations and Trends in Theoretical Computer Science*, vol. 3, pp. 93–263, 2009.

[12] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM*, vol. 44, no. 4, pp. 585–591, 1997.

[13] T. Kesselheim, A. Tönnis, K. Radke, and B. Vöcking, "Primal beats dual on online packing LPs in the random-order model," in *ACM STOC*, 2014, pp. 303–312.