DNS Does Not Suffice for MEC-CDN

Ke-Jou Hsu James Choncholas Ketan Bhardwaj Ada Gavrilovska
Georgia Institute of Technology
{nosus_hsu, jchoncholas3, ketanbj}@gatech.edu, ada@cc.gatech.edu

ABSTRACT

Mobile edge computing (MEC) can transform mobile networks into a new infrastructure tier for services requiring low response times, such as those providing content to emerging AR/VR, autonomous driving, and other types of applications. To be successful, the CDNs operating in this MEC infrastructure tier – MEC-CDNs – will need to ensure end user applications gain access to a cache server in a fast and accurate manner. This paper sheds light on the challenges that the current mobile DNS architecture poses toward achieving this goal, and presents ideas on how to re-architect the existing DNS architecture to enable CDNs to provide low-latency content delivery from the edge.

CCS CONCEPTS

• **Networks** → **Mobile networks**; • **Computer systems organization** → **Distributed architectures**.

KEYWORDS

edge computing; mobile network; DNS; CDN

ACM Reference Format:
Ke-Jou Hsu James Choncholas Ketan Bhardwaj Ada Gavrilovska . 2020. DNS Does Not Suffice for MEC-CDN. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks (HotNets '20), November 4–6, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3422604.3425931

1 INTRODUCTION

Mobile (or Multi-access) Edge Computing (MEC) presents transformative opportunities for mobile network operators to *cloudify* their networks, by leveraging computational infrastructure in the edge of their networks – at base stations – for hosting third-party workloads. This new infrastructure tier, by being closer to end users and devices, provides edge applications with ability to serve requests with lower latency and with lower traffic loads posed on the backhaul network.

One prime candidate for edge deployment are Content Distribution Networks (CDNs), with current response-time latencies ranging from 20 ms (best in class) to 300 ms [3, 47]. These times far exceed the sub 20 ms requirements of emerging workloads such as AR/VR [5, 26, 49], autonomous driving [50], and others [34]. Deploying CDNs at the edge also offers opportunities for location-aware, contextualized content delivery, by presenting different content from different edge locations based on context. Providing these capabilities with current CDN deployments is not possible.

In theory, deploying CDNs at the mobile network edges should suffice for delivering latency-sensitive content for satisfying the aforementioned scenarios. However, a key component in realizing any solution is the behavior of the DNS infrastructure which is responsible for routing incoming requests to a CDN cache server. In order for MEC-based CDNs to be successful – which we refer to as **MEC-CDNs** – the DNS service must provide the following:

- **P1** DNS requests **must be quickly resolved**, with latencies that are far below the application-level end-to-end latency requirements.
- **P2** DNS responses **must identify the nearest CDN server**, i.e., the cache server hosting the requested content and deployed in the edge location.

Without P1 and P2, CDNs will never be ready for deployment at the edge.

Prior work has already presented an abundance of evidence on the poor performance of DNS systems across different generations of mobile networks [46] and its impact on CDN performance. Concerning location-aware contextualization, modern CDN strategies for routing incoming requests cannot guarantee geographic proximity of the responding server to the origin of the request [48]. The request's origin is often obfuscated in current mobile networks including the client's IP address [35] (CDN servers see the public gateway's IP, not the end client's) and the geographic location of the incoming request (CDN servers infer the location of the public gateways using GeoIP lookup and that too with limited accuracy [18]). The Extended DNS Client Subnet feature (ECS) [39] is proposed as a way to solve the DNS localization problem. However, using ECS alone still requires network traversals outside of the MEC complex, and is shown to be susceptible to problems related to hidden resolvers [36]. Furthermore, ECS does not help identify a specific CDN cache server among multiple cache server instances which, for scalability reasons, are co-running at a MEC location.

Despite such known technical issues, solving them is non-trivial because of conflicting incentives of mobile network providers and CDN providers. While mobile network providers are more interested in load balancing and reusing IP address blocks across different customers, for CDNs the same practice makes geo-localizing IPs difficult with respect to the location of appropriate cache servers at their points of presence. Despite convergence of the two in some cases [31], introduction of new entities such as CDN brokers, has been shown to invalidate many other traditional assumptions, even excluding CDNs themselves in the cache server selection [45]. We posit that such opaqueness will continue to exist in 5G with MEC, unless there is a disruptive design change in how CDNs are deployed and, more importantly, how DNS is managed in the MEC-CDN ecosystem. In fact, very recent proposals from ETSI [12] and from the 3GPP working group on the integration of MEC and

HotNets '20, November 4–6, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8145-1/20/11…$15.00
https://doi.org/10.1145/3422604.3425931

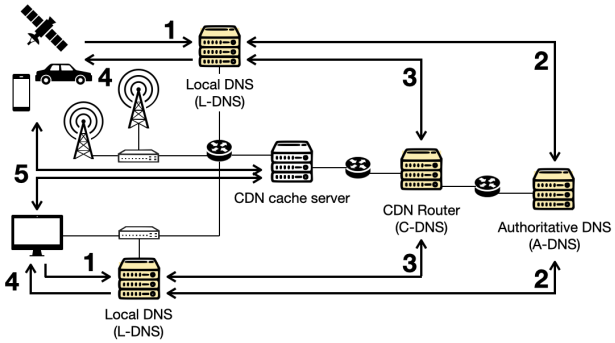


Figure 1: Steps between DNS and CDN when clients access CDN content via wired and mobile networks.

5G [33], acknowledge the limitations of the existing DNS architecture, and argue for a DNS component to be included at the MEC. The primary goals of these proposals are enabling seamless discovery of services deployed in the MEC platform, including during user mobility. However, the proposals do not focus on the unique aspects of hosting CDNs as MEC services. Furthermore, given their draft proposal form, these documents do not include evaluation of the proposed ideas and their practical impact is yet to be assessed.

In this paper, we further emphasize why the alignment of DNS *and* CDN deployments for and at MEC, respectively, is even more critical now. If they continue to operate without alignment, it is **infeasible** to achieve location-aware, low-latency cache localization that is needed for a MEC-CDN to deliver on its promises of sub 20 ms latencies. Summarizing, the contributions we make are:

- We present empirical evidence and a discussion of the reasons for the limitations of current DNS service deployments for achieving low latency (Section § 2);
- We present a set of design decisions that address the current limitations, and discuss the technical challenges that must be considered when developing a complete solution for MEC-CDN (Section § 3); and
- We present preliminary results, conducted on an end-to-end MEC-CDN deployment on a private 4G-LTE network testbed, realized with fully containerized commercial, off-the-shelf srsLTE-based RAN, NextEPC 4G LTE core, and the Apache Traffic Control (ATC) CDN. We demonstrate that the proposed design is practical and can result in up to 9× faster DNS resolution, thus providing drastic reductions in the access latency for content cached in MEC-CDNs, compared to current commercial CDN deployments (Section § 4).

2 DNS MAY KILL MEC-CDN

Prior work has established that the performance of CDNs over mobile networks is tied to the behavior of the DNS resolution process [35, 37, 40, 43, 46]. This is the case for MEC-CDN too. To understand it better, we start with a high level overview of the DNS–CDN interactions, backed by our initial experimental observations.

Q1: How should CDNs work?

Figure 1 shows the high level steps involved in a CDN access over two kinds of Internet connectivity: wired and mobile. (1) The client issues a DNS lookup query for a CDN domain based on the content URL. (2) The Local DNS (L-DNS) receives the request and resolves the domain via the root, top-level domain, and authoritative DNS (A-DNS). A-DNS responds with the CNAME for the CDN's name server. (3) L-DNS sends a DNS lookup for that CNAME. The CDN Router, a name server (C-DNS) operated by the CDN system, receives the request. Based on L-DNS's location, C-DNS returns the IP address of a cache server, which ideally is the nearest one to the L-DNS. (4) L-DNS responds to the client with the CDN cache server's IP address. (5) The client accesses the cached content from that CDN's cache server.

As shown in this sequence, a multi-layer hierarchy of DNS servers (marked in yellow in Figure 1) is involved in the process. Together, their goal is to capture the client's location (or context) and route the client request to a nearest CDN cache server which has the requested content. In order to offer best CDN service to applications, the end user must be able to access a cache server in a **fast** and **accurate** manner: (i) To shorten the networking delay, L-DNS and the cache server should be in a near location to clients (step 1, 4, and 5 in Figure 1). (ii) L-DNS and C-DNS need to perform fast domain name resolution. Part of the DNS resolution time in C-DNS is based on the CDN internal caching mechanisms around their server hierarchy, naming, indexing, content placement, cache miss policy, etc. (iii) C-DNS must pick a cache server which has the content and is nearest to the client. Aligning these expectations of a CDN system with the DNS requirements stated in **P1** and **P2**, we need to satisfy (i) and (ii) to solve **P1**, and ensure (iii) for **P2**.

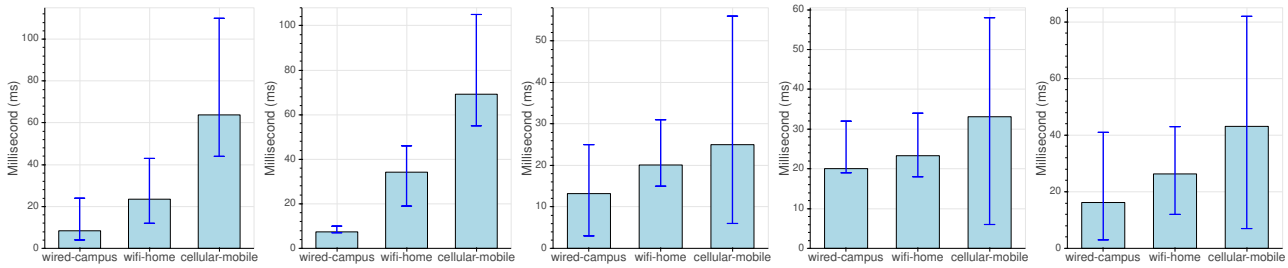
Online travel agency	Tested CDN domain name
Airbnb	a0.muscache.com
Booking.com	q-cf.bstatic.com
TripAdvisor	static.tacdn.com
Agoda	cdn0.agoda.net
Expedia	a.cdn.intentmedia.net

Table 1: The CDN domain name we tested for static web content (.img, .js, .css...etc).

Q2: How do CDNs work?

To investigate how well the high-level understanding of "how CDNs should operate" maps to real CDN deployments, accessed via different type of Internet connectivity, we performed a set of simple tests from end devices. We picked five popular travel agent websites owing to their popularity and known substantial use of CDNs. We were easily able to identify the content and its CDN domain by inspecting the source code using the preview feature in the Chrome browser, as most of the content is addressed using URLs that contain the CDN domain names in plain text. Table 1 lists the five websites used in our experiments, and their CDN domains.

We used the DNS lookup utility `dig` for the five CDN domains from the same device on three kinds of networks: wired campus network, wireless network through a home Wi-Fi router, and mobile network through a hotspot on a cellphone. All experiments are performed from the exact same geographic location. Figure 2 shows the average latency of the DNS queries for each of the domain names from Table 1. This query time corresponds to the sum of the latencies of steps 1, 3 and 4 in Figure 1; We observed that these websites directly use the CDN domain name for the cached content, therefore CNAME lookup from A-DNS (step 2) is never required. This is inline with expectations that for popular websites' CDN domains, the A records TTL never expires at L-DNS and the cached A records are used for lookup.



(a) Airbnb (b) Agoda (c) Booking.com (d) Expedia (e) TripAdvisor

Figure 2: DNS lookup latency for CDN domain URLs for three types of Internet connectivity. Each bar is based on at least 12 tests, only including the results from the 8th- to the 92th-percentile. The maximum and minimum are marked with error lines.

We further analyzed the DNS responses for each website. We found that regardless of the network connectivity, the requests for content from a fixed CDN domain URL issued from the same location, are routed to different cache servers. Figure 3 illustrates this by showing the distribution of responses across different cache servers for the different access configurations for the same five websites. For instance, only Amazon CloudFront provides cached objects in this domain for Booking.com (Figure 3c), but it has multiple CIDR ranges, with different distribution of the responses for the different types of the connectivity. In a different instance, TripAdvisor uses multiple CDNs, some which have multiple CIDRs for the same domain URL. In summary, there exists a non-trivial mapping of which server is chosen, and the process of how that decision is made is opaque to end users and sometimes to the CDN itself [45]. From the experiments, we make the following observations:

1. *Accessing the L-DNS in mobile networks incurs a substantially higher delay and higher response time variability*, as seen by the bars "cellular-mobile" in Figure 2. This performance degradation points to a potential delay incurred in the wireless network itself, the RAN software stack and the opaque deployment of cellular L-DNS [46]. *With 5G, while the latency incurred by the wireless hop is expected to be drastically reduced, a new design for L-DNS and tight integration with the RAN stack will still be required to reduce the end-to-end DNS query latency for CDNs.*
2. Across the different kinds of Internet connectivity, CDN requests are served from different cache servers, with different frequency, as shown in Figure 3. In other words, *although clients send requests from a similar geo-location, they are not guaranteed to access the content from the same set of cache servers.* This also leads to disaggregation of requests and may increase the cache miss rate.

One can easily tell how these observations are generalizable. The steps involved in CDN operation remain the same for any web content provider that uses a CDN service to store frequently-visited static content using static URLs. An obvious question to ask then, is how can CDN performance be guaranteed. To answer that, we need to better understand the ecosystem and its participants.

Q3: Who owns performance in MEC-CDN?

Even through, at a high level, the operation of DNS seems straightforward, the ecosystem involves many entities, which makes it difficult to implement changes in the DNS-CDN interactions. While each of the entities subsumes a single default role, as shown in Table 2, an entity can also take on multiple, or even all of of these roles. For instance, cellular providers are known to include DNS or CDN provider roles (e.g., Verizon [31]). Each of the entities contributes to the end-to-end performance of CDNs. For example, CDN brokers have been shown to cause shifts in the traffic patterns observed by a CDN, leading to performance and efficiency issues [45]. Adding MEC providers to the mix just makes the situation worse, particularly since there are many kinds of MEC service providers: a cellular provider acting as MEC provider [9], a cloud provider acting as a proxy for a cellular provider's MEC [19, 32], and standalone edge providers that have a footprint across many cellular providers [10, 20, 21].

Entity	Role
Cellular Providers	Operating RAN and cellular core network
CDN Providers	Providing content caches on CDN domains hosted on some server nodes
DNS Provider	Routing requests to closest CDN domain servers
Web Provider	Delivering web services that use CDNs to provide better services to end users
Cloud Provider	Providing server infrastructure to one or more of the above
CDN Brokers	Providing a consolidated service spanning multiple CDNs to CDN customers
MEC Provider	Providing MEC servers that host CDN domains

Table 2: Entities and roles in MEC CDN

Our experiments illustrate such inconsistencies as well. From Figure 3, we hypothesize that the load balancing decisions that route queries to cache servers, within a provider (Akamai, CloudFront and Fastly have multiple sites), or across multiple CDN providers (Airbnb, Expedia and TripAdvisor depend on two or more CDNs for a single cache domain), are related to cascading CNAMEs managed by other DNS providers (e.g., GoDaddy [15]). Moreover, some CDN services are deployed in a cloud and rely on the cloud provider's services, e.g., Amazon Route 53 [2]. A cloud provider can also offer a CDN service, like Amazon CloudFront [1]. All of these various CDN deployment choices make it intractable to point out the "invisible performance owners" who could guarantee CDN performance.

Regardless of how many entities are involved in the CDN ecosystem or how their interactions are managed today, what remains critical for MEC-CDN to deliver on ultra low-latency is for a DNS provider to quickly respond with appropriate IPs for CDN domains.

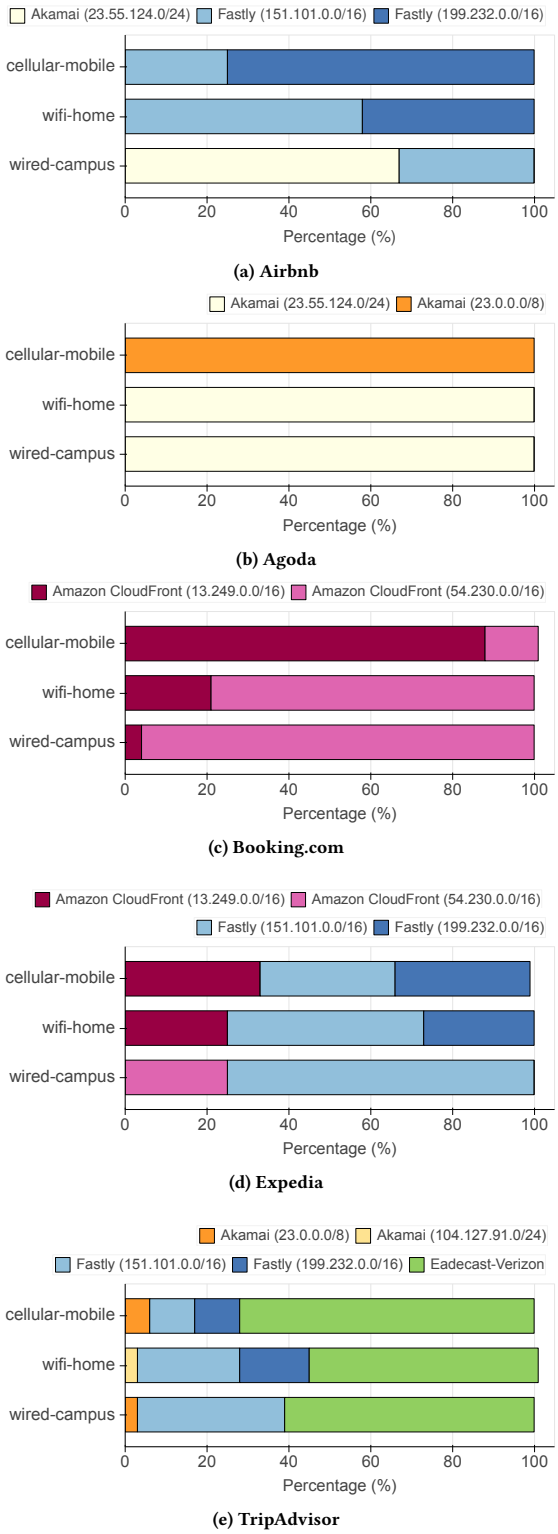


Figure 3: Distribution of DNS responses among different cache servers. Each bar represents the percentage of responses in step 4 of Figure 1 identifying a given cache server.

3 RE-DESIGN OPPORTUNITIES

Prior efforts have focused on optimizing individual steps in the operation of CDNs [42] or on improving the performance of a particular role in the CDN ecosystem [11, 14]. Our goal, instead, is to explore opportunities afforded by re-designing DNS – CDN interactions in MEC-CDN, in a way that delivers on the low latency requirements of MEC, without adding to the complexity of the CDN ecosystem. To realize MEC-CDN deployments with MEC latency benefits, we explore a design that aims to combine two different sets of functionality – the DNS query for CDN domain resolution and the lookup of a cache server which can provide the requested content – in a single hop operation that can be fully contained at MEC. Furthermore, we aim to ensure that the design is in line with current trends towards fully containerized 5G RAN [41], 5G Core Network and MEC [13].

However, this design decision is non-trivial. It not only impacts the way CDNs operate and how a cache can be accessed by end users, as shown in Figure 4, but also requires role modifications of the entities in the MEC-CDN ecosystem. Furthermore, it requires additional functions that need to be carried out for seamless operation from the end user's perspective. Below we discuss how the two goals for MEC-CDN that we outlined earlier are achieved through several design decisions, thus making CDNs edge-ready.

P1: Finding a cache quickly.
The first step in finding a cache quickly is to refer to the nearest DNS server. One critical design decision is the question where is DNS deployed? Today, hierarchical DNS [46] service is deployed behind the cellular core network either by cellular providers [27], cloud providers [2, 6] or CDN providers [7, 17]. The hierarchical nature of DNS increases the resolver distance and adversely impacts DNS resolution latency. None of these deployment architectures are suitable for MEC-CDN to operate within a latency envelope less than 20 ms, as also demonstrated in our preliminary results (Figure 5). Even considering the recent proposals from ETSI and 3GPP [12, 33], the specific steps they outline continue to involve hops to external A-DNS or application-specific components (equivalent to the C-DNS routing element in Figure 1), and introduce additional complexity related to virtual IP translation, HTTP redirection, etc., and as such will not be adequate for resolving MEC-CDN accesses with latencies commensurate of what is expected from MEC.

Fortunately, the MEC orchestrator is a core component in all of the network function virtualization (NFV) approaches that enable deployment of containerized MEC in a containerized vRAN [23, 25, 33]. The information needed to service DNS requests in the MEC – which CDN domains (or MEC applications) are deployed at a particular MEC location, and their public facing IP addresses – is readily available with the MEC orchestrator by design, as part of the MEC orchestrator's dedicated, internal DNS. The orchestrator uses this internal DNS for service discovery and management actions for VNFs. An example of such DNS is CoreDNS in a Kubernetes orchestrated virtualized network [24].

Intuitively, the same internal DNS instance can be re-purposed and used directly to quickly resolve a CDN domain upon a request from a client, which (on a hit) would eliminate the need for hierarchical lookups. However, exposing an internal DNS publicly to clients increases the attack surface for the vRAN itself by exposing the

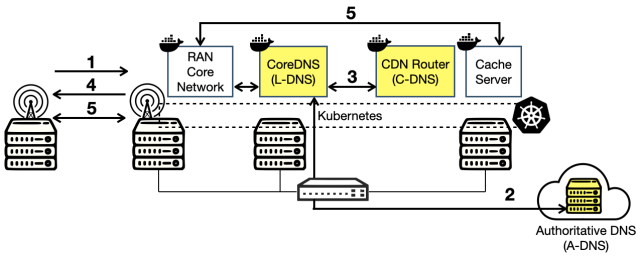


Figure 4: The proposed MEC-CDN system and its CDN processing steps aligned to Figure 1.

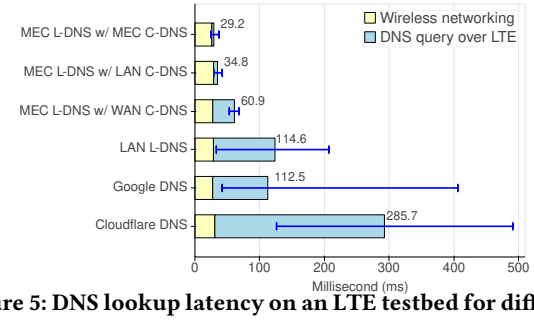


Figure 5: DNS lookup latency on an LTE testbed for different local resolvers and for MEC-CDN; Bars show the average values, error lines show the maximum and minimum.

vRAN IP namespace. To avoid that, we first run a **split-namespace DNS** as part of the MEC/vRAN infrastructure. This comprises at least two DNS namespaces: one namespace instance dedicated for internal VNFs, and another namespace instance for publicly visible IPs, i.e., for MEC-CDN. The publicly visible namespace is populated when a MEC-CDN instance is deployed.

To connect to the MEC L-DNS, when an end user connects to a particular base station, its target DNS is switched to that of the MEC DNS. This can be performed either as part of the cellular hand-off process, or explicitly, through a user-level MEC specific app that chooses the edge location [20] or as suggested recently by ETSI [12]. With that, it is now possible for end users to quickly find a CDN domain right at the first hop away from them, adhering to the MEC latency envelope by eliminating all of the hierarchical lookup delays.

Ideally, end user DNS requests will only refer to this DNS for discovering MEC-based services. The DNS queries for other services should continue to be handled by the users' or the provider's default L-DNS. A simple workaround that adds complexity on the user side would have the MEC DNS ignore queries not related to MEC-CDN, and have DNS requests be multicast to both MEC DNS and the network's L-DNS, or even be forwarded to L-DNS on timeout from MEC DNS. As shown in our preliminary results, this can be acceptable, given that the MEC DNS resolution can be achieved up to 3× faster compared to when using the existing network L-DNS, thus adding only a small overhead to CDN accesses for non-latency-critical content (i.e., content which otherwise is not hosted at the MEC). In case when MEC DNS is slower, end users will observe only a degradation but not unavailability of DNS resolution. In that sense, MEC CDN provides best effort guarantees, and also affords potential prevention of DoS attacks on the infrastructure. The MEC orchestrator, which has access to monitoring statistics of the ingress network load to the MEC DNS, can simply switch (or only unicast) to the provider's L-DNS during high ingress (above a threshold), or deploy other more sophisticated mitigation policies.

P2: Finding the right cache.
Even when a CDN domain is found quickly, to guarantee end-to-end MEC-CDN performance, it remains critical to ensure the end user accesses the right cache server. Ensuring this requires that the URLs of static content must point to domains available where end users are located.

In CDNs, the responsibility for identifying the cache server lies with the CDN router, or C-DNS in Figure 1. The C-DNS is responsible for identifying the cache server and responding with its IP address based on factors such as the origin of the request, load balancing strategy, etc. To make it possible to identify the right edge server, with low latency, it is then necessary to **collocate C-DNS** at MEC. By placing a C-DNS at MEC, it can have a scope limited only to the cache server instances at the edge location. As such, we allow it to find the right cache instance (provided the content is available) and to do so more quickly, because the content server is implicitly available and there are (likely) fewer cache servers to be considered. In cases where the content is not available at MEC-CDN, C-DNS simply returns the address of another C-DNS running at a different CDN tier, e.g., a mid-tier running alongside the mobile network core, or a far-tier running in the cloud, accessible over WAN.

To fully realize the requirements in P2, content mapping to MEC IP addresses can be achieved in straightforward manner by using separate DNS plugins for handling the two namespaces differently. What this enables fundamentally is that it combines the L-DNS lookup with a C-DNS lookup that is carried out at the first hop, in the MEC, eliminating additional C-DNS lookup(s).

Moreover, this design also addresses another practical concern in deploying MEC applications, i.e., it reduces the number of public IP addresses needed at MEC [16]. Today, most CDNs either use their own IP allocations or the ones provided by the cloud infrastructure. With the MEC provider being responsible for providing such IP addresses for all MEC applications, the required number of public IPs needed will be huge. The proposed design can help promote reuse of public IPs by assigning the same public IP for CDN domains of the many CDN customers. At a high level, what this enables is spatial reuse of IP addresses available at MEC akin to spatial reuse of spectrum in 5G. However, this does require appropriate routing support to be in place; this can be easily provisioned through the MEC orchestrator which already controls the routes among constituent containers, be it network functions or MEC applications.

4 PROTOTYPE AND EARLY RESULTS

Prototype Implementation
To illustrate the feasibility of MEC-CDN, we deploy our own RAN testbed with CDN (Figure 4). We used two USRP B200mini [30] and four Intel i7 machines. One machine with an USRP acts as a mobile end user, another machine with the other USRP forms the radio base station, and the remaining machines represent the collocated MEC servers and a distributed 4G-LTE core network. With the exception of the machine acting as a UE, the remaining three machines are collocated at the edge of network. We use *NextEPC* [22] for the

core network functions, *srsLTE* [29] for the RAN eNB and UE, and *Apache Traffic Control* (ATC) [4] for the MEC-CDN. All of the collocated radio network components and the MEC-CDN application are containerized and managed by *Kubernetes* (k8s) [44].

CoreDNS is the default DNS server of k8s, in charge of service discovery for any k8s computing unit. In the MEC-CDN prototype, it works as the cellular L-DNS for mobile users. C-DNS in Figure 4 is the *Traffic Router* in ATC. It responds to client's request with the IP address of a cache server, depending on the requested content, the cache servers' configurations and their availability at the edge. CoreDNS (L-DNS) manages the k8s domain and Traffic Router (C-DNS) manages the CDN domain. With respect to deploying such DNS servers with different ownerships, i.e., the cellular network L-DNS and the CDN's DNS, we first assign C-DNS a fixed cluster IP using k8s Service [28]. This ensures the C-DNS availability regardless of any scaling event. Next, we update the configuration of L-DNS with the sub-domain and upstream server [8] to ensure that L-DNS redirects queries for this CDN domain to C-DNS.

The external gateway components of the RAN would naturally forward internal-facing packets to other k8s services; as such, the DNS query packets are routed to CoreDNS. This benefits both the MEC platform provider and the application provider since none of the k8s hosts need to be explicitly exposed (as discussed in [12, 33]) in order to deliver MEC services. This design does not impose any restrictions on the developers' use of domain names at MEC. Specifically, in our implementation, when the UE requests content from `video.demo1.mycdn.ciab.test`, the DNS request goes via the wireless connection to the k8s-managed vRAN. The P-GW (Packet Data Network Gateway in RAN) directs the request to our MEC DNS, CoreDNS in k8s, which recognizes the request is for ATC, and passes it to C-DNS, the Traffic Router of ATC.

Does the re-design of MEC-CDN enable faster DNS lookup? In Figure 5, we evaluate the impact of the MEC-CDN techniques on DNS resolution latency by comparing it with several other possible DNS deployments. For the second and third bars, only L-DNS is collocated at MEC. The CDN cache server is located at MEC as well, but the C-DNS is outside of the MEC k8s cluster, connected via LAN (best case) or WAN. These deployments match the most recent proposals by ETSI [12] and 3GPP [33]. The remaining three cases show the performance achievable with current L-DNS deployments: the mobile L-DNS connected via LAN behind the core network, the cloud-based public DNS services from Google, chosen due to its wide adoption [38], and the one from Cloudflare, as an example of a CDN DNS service. For each bar, we break down the DNS lookup into two parts: (i) the delay due to the wireless communication of the mobile network, i.e., between the UE (client) and the P-GW, and (ii) the time spent in the DNS resolvers, the core network components, and any up/downlink communication delay. We perform the measurements using both `dig` from the client side and `tcpdump` at P-GW to track the DNS request packets.

We make several observations. First, the MEC-CDN solution, which deploys both L-DNS *and* C-DNS in the MEC, far outperforms all other currently available or proposed DNS resolution options. MEC-CDN results in up to 9× lower resolution latency compared to the existing solutions which do not deploy L-DNS in the MEC. Second, concerning the recent proposals for integrating DNS and MEC, Figure 5 shows that without collocating the CDN resolver at MEC as well, even if the requested content is present in the MEC-based CDN cache server, the DNS request would incur the additional hop to the C-DNS in the overall lookup. In MEC-CDN, this is eliminated and the DNS resolution can be fully contained within the MEC. Third, on the current MEC testbed, a dominant component of the MEC L-DNS time is the wireless LTE latency (approx. 10 ms one way). Future 5G deployments will drastically reduce this time, resulting in even greater end-to-end boost for MEC-CDN. Finally, other than MEC-CDN, only the ideal scenario of C-DNS being deployed outside but on the same LAN as MEC, makes it possible to serve a DNS request with sub-20 ms end-to-end latency. The 5ms lower latency of MEC-CDN, compared to this ideal setting, can represent a significant portion of the latency headroom for important classes of emerging applications, such as AR/VR, automated transportation, etc. When C-DNS is deployed farther in the WAN, even the best-case latencies exceed the response time requirements, making it unsatisfiable for serving latency-critical content from the edge.

We also evaluated the use of the EDNS Client Subnet feature (ECS) [39], implemented by enabling ECS support at L-DNS and C-DNS for the first three deployment scenarios above. ECS changed the measurements by 1.01x, 1.08x and 0.95x, respectively, i.e., using ECS may even increase DNS resolution time. In these experiments the DNS query was always correctly resolved to the appropriate CDN cache server at the MEC.

5 SUMMARY

We identify a critical barrier to low-latency MEC-CDN solutions – the current DNS architecture. We present two practical design decisions that provide for DNS queries to be resolved at the edge, *quickly*, with latencies well below the end-to-end latency requirements of emerging latency-critical applications, and *accurately*, responding with IP addresses of cache servers deployed at edge.

One of the non-intuitive, but important, advantages of the proposed design for MEC providers is that, it makes it feasible to deploy CDNs at MEC without requiring any public-facing IP addresses dedicated for MEC services. With the re-design of MEC-CDN, instead of exposing L-DNS, C-DNS, and the CDN cache servers and their host IPs, mobile clients interact with CDNs by merely using the Kubernetes cluster IPs associated with the MEC L-DNS.

The above benefits are made feasible due to the end-to-end orchestration of the containerized RAN, core network, MEC and CDN, through a single logically centralized orchestrator system. This is well aligned with the roadmaps adopted by all major network operators for 5G and beyond, making the proposal practical. We provide initial experimental results that demonstrate the feasibility of the solution, and highlight a number of technical issues that need to be solved, so as to deliver a robust MEC-CDN architecture.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd, K. K. Ramakrishnan, for their valuable feedback toward improving the paper. The work on this project is supported through funding grants, gifts, and equipment donations by Intel, VMware, NSF award CNS-1909769, and the Applications Driving Architectures (ADA) Research Center, a JUMP Center co-sponsored by SRC and DARPA.

REFERENCES

[1] [n.d.]. Amazon CloudFront. https://aws.amazon.com/cloudfront/.
[2] [n.d.]. Amazon Route 53. https://aws.amazon.com/route53/.
[3] [n.d.]. Analysing Global CDN Performance. https://labs.ripe.net/Members/emirb/analysing-global-cdn-performance.
[4] [n.d.]. Apache Traffic Control. https://traffic-control-cdn.readthedocs.io/en/latest/index.html.
[5] [n.d.]. Cloud AR/VR Whitepaper. https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/.
[6] [n.d.]. Cloud DNS | Google Cloud. https://www.verizondigitalmedia.com/free-trial/dns-services/.
[7] [n.d.]. Cloudflare DNS | Managed DNS | Cloudflare. https://www.cloudflare.com/dns/.
[8] [n.d.]. Configuration of Stub-domain and upstream nameserver using CoreDNS - Kubernetes. https://kubernetes.io/docs/tasks/administer-cluster/dns-custom-nameservers/.
[9] [n.d.]. Deutsche Telekom Completes World's First Public Mobile Edge Network, Powered By MobiledgeX Edge-Cloud R1.0. https://mobiledgex.com/press-releases/2019/02/19/deutsche-telekom-completes-worlds-first-public-mobile-edge-network-powered-by-mobiledgex-edge-cloud-r1-0.
[10] [n.d.]. EdgeConneX | Hyperlocal to Hyperscale Data Centers. https://www.edgeconnex.com.
[11] [n.d.]. EdgeNet Project. https://edge-net.org.
[12] [n.d.]. Enhanced DNS Support towards Distributed MEC Environment. https://www.etsi.org/images/files/ETSIWhitePapers/etsi-wp39-Enhanced-DNS-Support-towards-Distributed-MEC-Environment.pdf.
[13] [n.d.]. ETSI Mobile Edge Computing. https://www.etsi.org/technologies/multi-access-edge-computing.
[14] [n.d.]. Fastly | The edge cloud platform behind the best of the web. https://www.fastly.com.
[15] [n.d.]. GoDaddy: Domain Names, Websites, Hosting & Online. https://godaddy.com.
[16] [n.d.]. Intel Engineer: Much Left to Solve Before Cloud and Edge Can Become One. https://www.datacenterknowledge.com/edge-computing/intel-engineer-much-left-solve-cloud-and-edge
can-become-one.
[17] [n.d.]. Introducing Akamai Edge DNS. https://blogs.akamai.com/2020/03/introducing-akamai-edge-dns.html.
[18] [n.d.]. MaxMind: IP Geolocation and Online Fraud Prevention. https://www.maxmind.com/.
[19] [n.d.]. Microsoft, AT&T create edge compute zones. https://www.fiercetelecom.com/telecom/microsoft-at-t-create-edge-compute-zones.
[20] [n.d.]. MobiledgeX: Edge Computing. https://mobiledgex.com.
[21] [n.d.]. Multi access Edge Computing Pioneer - Saguna. https://www.saguna.net.
[22] [n.d.]. NextEPC. https://nextepc.org.
[23] [n.d.]. ONAP. https://www.onap.org.
[24] [n.d.]. ONAP and Kubernetes Mashed Together for App Portability. https://www.sdxcentral.com/articles/news/onap-and-kubernetes-mashed-together-for-app-portability/2018/03/.
[25] [n.d.]. OpenNESS - Open Network Edge Services Software. https://www.openness.org.
[26] [n.d.]. Problem Statement: Transport Support for Augmented and Virtual Reality Applications. https://tools.ietf.org/id/draft-han-iccrg-arvr-transport-problem-00.xml.
[27] [n.d.]. Reliable DNS | DDoS Protection | The Verizon Media Platform. https://www.verizondigitalmedia.com/free-trial/dns-services/.
[28] [n.d.]. Service - Kubernetes. https://kubernetes.io/docs/concepts/services-networking/service/.
[29] [n.d.]. srsLTE. http://www.softwareradiosystems.com.
[30] [n.d.]. USRP B200mini. https://www.ettus.com/all-products/usrp-b200mini/.
[31] [n.d.]. Verizon Media Platform. https://www.verizondigitalmedia.com/our-company/about/.
[32] [n.d.]. Verizon partners with AWS to bring more power to its 5G edge. https://www.fiercewireless.com/5g/verizon-partners-aws-to-bring-more-power-to-its-5g-edge.
[33] 2020. *Technical Specification Group Services and System Aspects; Study on enhancement of support for Edge Computing in 5G Core network (5GC) (Release 17).* Technical Report 23.748 V1.0.0 (2020-09). 3rd Generation Partnership Project (3GPP).
[34] Telefonica Alpha. [n.d.]. The Edge: Evolution or Revolution? http://acm-ieee-sec.org/2017/Edge%20Computing%20SEC%20Keynote%20Oct%202017%20Pablo%20Rodriguez.pdf.
[35] Walid Benchaita, Samir Ghamri-Doudane, and Sébastien Tixeuil. 2016. Stability and optimization of DNS-based request redirection in CDNs. In *Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, January 4-7, 2016*. ACM, 11:1–11:10. https://doi.org/10.1145/2833312.2833455
[36] M. Calder, X. Fan, and L. Zhu. 2019. A Cloud Provider's View of EDNS Client-Subnet Adoption. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*. 129–136.
[37] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, Kenjiro Cho, Kensuke Fukuda, Vivek S. Pai, and Neil Spring (Eds.). ACM, 531–537. https://doi.org/10.1145/2815675.2815717
[38] Patricia Callejo, Rubén Cuevas, Narseo Vallina-Rodriguez, and Ángel Cuevas Rumin. 2019. Measuring the Global Recursive DNS Infrastructure: A View From the Edge. *IEEE Access* 7 (2019), 168020–168028.
[39] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. 2016. Client Subnet in DNS Queries. *IETF RFC7871, May* (2016). https://tools.ietf.org/html/rfc7871
[40] Matteo Dell'Amico, Leyla Bilge, Ashwin Kumar Kayyoor, Petros Efstathopoulos, and Pierre-Antoine Vervier. 2017. Lean On Me: Mining Internet Service Dependencies From Large-Scale DNS Data. In *Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, December 4-8, 2017*. ACM, 449–460. https://doi.org/10.1145/3134600.3134637
[41] Juuso Haavisto, Muhammad Arif, Lauri Lovén, Teemu Leppänen, and Jukka Riekki. 2019. Open-source RANs in Practice: An Over-the-air Deployment for 5G MEC. *CoRR* abs/1905.03883 (2019).
[42] Jeffrey Helt, Guoyao Feng, Srinivasan Seshan, and Vyas Sekar. 2019. Sandpaper: mitigating performance interference in CDN edge proxies. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, SEC 2019, Arlington, Virginia, USA, November 7-9, 2019*, Songqing Chen, Ryokichi Onishi, Ganesh Ananthanarayanan, and Qun Li (Eds.). ACM, 30–46. https://doi.org/10.1145/3318216.3363313
[43] Hadrien Hours, Ernst Biersack, Patrick Loiseau, Alessandro Finamore, and Marco Mellia. 2016. A study of the impact of DNS resolvers on CDN performance using a causal approach. *Comput. Networks* 109 (2016), 200–210. https://doi.org/10.1016/j.comnet.2016.06.023
[44] Kubernetes. 2018. Production-Grade Container Orchestration. https://kubernetes.io.
[45] Matthew K. Mukerjee, Ilker Nadi Bozkurt, Devdeep Ray, Bruce M. Maggs, Srinivasan Seshan, and Hui Zhang. 2017. Redesigning CDN-Broker Interactions for Improved Content Delivery. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'17)*. Association for Computing Machinery, New York, NY, USA, 68–80. https://doi.org/10.1145/3143361.3143366
[46] John P. Rula and Fabian E. Bustamante. 2014. Behind the Curtain: Cellular DNS and Content Replica Selection. In *Proceedings of the 2014 Internet Measurement Conference, IMC 2014, Vancouver, BC, Canada, November 5-7, 2014*. 59–72. https://doi.org/10.1145/2663716.2663734
[47] Sipat Triukose, Zhihua Wen, and Michael Rabinovich. 2011. Measuring a commercial content delivery network. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar (Eds.). ACM, 467–476. https://doi.org/10.1145/1963405.1963472
[48] Qiang Xu, Junxian Huang, Zhaoguang Wang, Feng Qian, Alexandre Gerber, and Zhuoqing Morley Mao. 2011. Cellular data network infrastructure characterization and implication on mobile content placement. In *SIGMETRICS 2011, Proceedings of the 2011 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, San Jose, CA, USA, 07-11 June 2011 (Co-located with FCRC 2011)*, Arif Merchant, Kimberly Keeton, and Dan Rubenstein (Eds.). ACM, 317–328. https://doi.org/10.1145/1993744.1993777
[49] Wenxiao Zhang, Bo Han, and Pan Hui. 2017. On the Networking Challenges of Mobile Augmented Reality. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. https://doi.org/10.1145/3097895.3097900
[50] Siyu Zhou, Prasad Prakash Netalkar, Yanan Chang, Yang Xu, and H. Jonathan Chao. 2018. The MEC-Based Architecture Design for Low-Latency and Fast Hand-Off Vehicular Networking. In *88th IEEE Vehicular Technology Conference, VTC Fall 2018, Chicago, IL, USA, August 27-30, 2018*. IEEE, 1–7. https://doi.org/10.1109/VTCFall.2018.8690790