(How Much) Can Edge Computing Change Network Latency?

Lorenzo Corneo*, Nitinder Mohan†, Aleksandr Zavodovski*, Walter Wong§, Christian Rohner*, Per Gunningberg*, Jussi Kangasharju§
*Uppsala University, Sweden
†Technical University Munich, Germany
§University of Helsinki, Finland

***Abstract*—Edge computing aims to enable applications with stringent latency requirements, e.g., augmented reality, and tame the overwhelming data streams generated by IoT devices. A core principle of this paradigm is to bring the computation from a distant cloud closer to service consumers and data producers. Consequentially, the issue of edge computing facilities' placement arises. We present a comprehensive analysis suggesting where to place general-purpose edge computing resources on an Internet-wide scale. We base our conclusions on extensive real-world network measurements. We perform extensive traceroute measurements from RIPE Atlas to datacenters in the US, resulting in a graph of 11K routers. We identify the affiliations of the routers to determine the network providers that can act as edge providers. We devise several edge placement strategies and show that they can improve cloud access latency by up to 30%.**

I. Introduction

Over the past decade, edge computing has emerged as a compellingly sounding solution for improving and enabling many next-generation networked applications. The excitement behind this computing domain majorly stems from its ability to improve overall latency by processing application services on devices installed close to the end-user. By doing so, edge servers naturally enable latency-sensitive applications, such as virtual reality, augmented reality, live video analytics, robotic control [1]–[3], etc. The capability to offer cloud-like services closer to the clients has ushered in a new-age revolution in industries like communication, medical, automobiles, etc.

While the utility and capability of edge computing to disrupt the technology market are unquestioned, the placement and availability of edge servers over the network is still an open problem plaguing the edge community. There are multiple possibilities for placements. Early advocates envisioned a world of user-controlled mobile devices that opportunistically form processing pools for short-lived applications [4], [5]. Industrial standardization initiatives, e.g., multi-access edge computing (MEC), suggest edge infrastructure to be a component of the ISPs [6]. Also, cloud providers are extending their existing networks by deploying compute servers at their point-of-presence. Content Delivery Network (CDN) providers have widespread storage servers that can also host edge computations [7], [8].

While simultaneous efforts from multiple interested parties may help popularize the capabilities of edge computing, we argue that together these deployments will not be able to fully harness the capabilities of the edge. One reason is probably that their launch strategies are often driven by competition for market dominance that may hinder interoperability in using edge servers. Despite the growing popularity of the edge within the research community, relatively little attention has been paid to understand the distribution of network latency between routers from a user device to public cloud providers. It is commonly believed that many latency-sensitive applications at clouds would benefit from running at edge servers close to the consumer instead [9].

In this paper, we focus on shared edge computing infrastructures, similar to the ones already employed by cloud providers. Hence, we explore the potential of reducing the latency of public cloud services by hypothetically placing edge servers at various routers along the path between users and different cloud providers. To this purpose, we conduct large-scale Internet `traceroute` measurements leveraging the RIPE Atlas platform [10] within the US, where we target 30 datacenters operated by *seven* major cloud providers. In addition to the usual user-to-cloud "vertical" traces, we run `traceroute` between all vantage points to get a broader knowledge of the user-serving network. This will give us additional "horizontal" paths that will complement the "vertical" ones. The collected dataset is publicly available at [11].

We evaluate various edge placement strategies and our results reveal that edge computing could bring a latency improvement between 6% to 30% with respect to the actual cloud access latency. More interestingly, we find that many "horizontal" paths we discovered can, in fact, deliver better cloud latency compared to the regular routing path, yielding latency gains of up to 40%. Our contributions are as follows:

1) We provide a large-scale latency study using the RIPE Atlas platform and `traceroute` from 900+ vantage points to seven major cloud providers – totaling 30 datacenters in the US.
2) We attribute the owners of each router. This gives an insight into potential providers of edge services, i.e., the ISPs and cloud providers that have the pervasiveness and router scale for an Internet-scale launch.
3) We evaluate several different edge placement strategies and find that they can reduce cloud access latency by up to 30% in some cases. However, the absolute values of

ISBN 978-3-903176-39-3© 2021 IFIP

the reductions remain on the order of few milliseconds.

II. Related Work

Cloud access latency has been an active research area for a long time, with [12] as one of the comprehensive studies, including OS latency and communication bandwidth. Recently, we conducted global studies on cloud reachability, with emphasis on access latency [13], [14]. In this work, we augment the aforementioned works with edge placement to reduce communication latency to computing resources. This is related to the large corpora of cache and CDN placement research. The foundational work by Krishnan et al. [15] characterizes the placement problem to be intractable in the general case, although giving algorithmic solutions for several restricted variants. Qiu et al. [16] investigate the issue in the context of CDNs and suggest a number of algorithms, which are essentially approximations of either facility location or K means problems, which are both NP-hard. In [17], the performance of CDN is enhanced by tightening cooperation with ISPs. Benkacem et al. [18] develop a theoretical framework for VNFs placement, which balances between two optimization targets, namely, cost and QoE. Concentrating on IoT needs, [19] utilizes information-centric networking, therefore differing significantly from our setting. Liu et al. [20] devise both centralized and decentralized placement algorithms to operate in fog radio access networks, finding their performance to be approximately equal. The methodology to harness network topology data for better CDN replica placement was introduced in [21]. Compared to our work, optimizing for communication latency to computing resources, cache placement strategies balance latency related to cache hits (equivalent to our scenario), and cache misses, which involve a significant latency to the data source.

Also, the edge research community has become active in placement issues. Wang et al. [22] develop a combinatorial optimization algorithm focusing on service entity placement for VR applications. For MEC environments, Xu et al. [23] offer an algorithm based on Lyapunov optimization and Gibbs sampling. Gao et al. [24] optimize placement in MEC environments further by taking into account network performance. Once again, such works' objectives are distant from ours as they tackle the placement of software services on hardware that is already deployed and available. However, this paper's goal is to explore the possible outline of in-network edge computing deployment and assess the latency gains that computational facilities could bring to end-users when compared to the already deployed cloud infrastructure.

III. Measurement Methodology

The focus of our work is to provide a better understanding of user to cloud connectivity from the network edge. We consider the network edge to begin with the last set of routers with public IP addresses located after the probes, thus excluding LAN devices. As we consider edge deployment in shared network infrastructure, this best corresponds to that point of view. While there are several datasets publicly available that attempt to map the Internet connectivity and routing at large [25], [26], we found them limiting for our study for several reasons. *Firstly*, existing projects primarily focus on mapping the entire IP address space rather than targeted measurements towards cloud end-points, which includes routes within datacenters. *Secondly*, the number and deployment location of probes used in these projects do not represent the user connectivity at the network edge. For example, CAIDA Ark project only hosts 52 probes in the US, the majority of which are hosted by network providers and educational institutes.

In this work, we fill this research gap by launching large-scale `traceroute` measurements towards datacenters of *seven* prominent cloud providers from RIPE Atlas platform [10]. `traceroute` provides information about the path between probes and datacenters, as well as the per-hop latency along the path. We process the collected data to build hop-centric and latency-centric network graphs describing user[1] to cloud connectivity. While our methodology can be applied to any network, we focus our study towards the US as it has the largest density of cloud datacenters and is an active area when it comes to deploying new network protocols and edge infrastructures [27]. Also, RIPE Atlas has a large number of probes in the US, enabling us to get a very dense network for our measurements.

A. Data collection

Vantage Points. We select vantage points for our measurements from RIPE Atlas [10], a *de-facto* standard, and well-established platform in the Internet measurements community.

RIPE Atlas is a globally distributed Internet measurements platform that is used extensively for reachability, connectivity, and performance studies. The platform provides thousands of small hardware probes connected to the Internet in a variety of installation environments, ranging from home networks to managed network core. Users can perform active network measurements, e.g., `traceroute`, from probes to end-points of their choice.

Despite Atlas's dense deployment nature, a large majority of the probes are hosted by cloud operators (CO) and network operators (NO), which allows them to monitor their network reachability from outside [28]. These probes do not reflect the connectivity from the network edge and have the potential to add bias to our measurements. Therefore, we filter out all the probes that are clearly installed in privileged locations, e.g., datacenters, from their user-defined tags [29] `datacentre`, `us-east*`, `us-west*`, `gcp` and `aws`. As these tags are user-contributed, they may be incorrect; we have attempted to verify some of them manually and they seem largely accurate, so we do not see this as a major concern. After filtration, we selected 934 probes scattered across 209 different networks (ASes) for our measurements. As we focus on the US, all the probes that ended up being selected were also from the US.

A further point to note is that RIPE Atlas probes are in the fixed network. While some probes may have wireless links in

[1]Despite statistical and operational differences, we use the terms "probes" and "users" interchangeably in this paper.

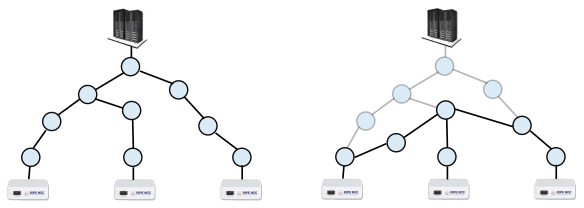

(a) *Probe-to-Cloud* (b) *Probe-to-Probe*

Fig. 1: Example of different network topologies discovered through our measurements to the cloud and through the probes.

their connectivity, the bulk of them are fully wired. For our study this is not an issue, since edge servers would most likely be placed in the fixed network and not on mobile nodes. In an actual deployment with wireless last-mile clients, the overall latency reduction would have the same absolute value that we observe, but the relative improvement would depend on the performance of the wireless link.

End-Points. Our `traceroute` measurements are divided into two parts – *probe-to-cloud* and *probe-to-probe* – one with the purpose to analyze the paths to the cloud servers, the other to explore additional possible edge locations close to the users.

1) Probe-to-Cloud: The goal of the Probe-to-Cloud measurement is to discover the network topology that carries traffic from the network edge (probes) to the cloud. We target 30 datacenters operated by *seven*[2] different cloud providers in the US, motivated by their popularity and effective coverage in the country. Specifically, we ran `traceroute` queries to public VMs hosted by CloudHarmony [30] in all chosen datacenters. The topology emerging from measurements from the probes to a particular datacenter is a tree with the datacenter as root and the probes as leaves (see Fig. 1a). Consequently, the measurements to all datacenters result in a set of trees.

2) Probe-to-Probe: While our *probe-to-cloud* measurements provide a useful network map that converges towards cloud datacenters, it provides limited information about the complete network topology. In order to further identify the network connectivity at the network edge, and to discover additional in-network routers that are possibly closer to the end-users, we launch *mesh-based* `traceroute` measurements between all probes in our dataset. Given the number of probes, pairwise measurements result in a large set of paths and related latency information. This knowledge serves to identify additional potential nodes to host edge servers and thereby provide lower access latency for the users. Fig. 1b illustrates the additional paths discovered by *Probe-to-Probe* measurements.

Ownership resolution. We supplement our router-level topology by augmenting it with the organizations in-charge of operating them. This allows us to better understand whether future edge server deployments are more convenient as a natural expansion of the cloud, or as capillary installation (as close as possible to end-users). To achieve this, we query the owners of the in-network routers we encounter in our `traceroute` measurements from a public `whois` database. We manually cluster the owners into three organization categories – *cloud operators* (CO), *network operators* and ISPs (NO), and *non-categorized* (NC). For instance, in our dataset, CO includes the operators of our 30 end-points while NO includes major US network operators such as AT&T, Comcast, etc. Finally, NC includes all the owners that do not belong to any of the previous categories, e.g., Internet Exchange Points (IXPs).

[2]Amazon, Microsoft, Google, DigitalOcean, Linode, Vultr and Alibaba.

B. Network graph generation

In this section, we explain how we sanitize our measurements and generate the network graphs for the hop and latency analysis. Before doing so, we give a primer about IP aliasing and how it affects our data collection.

Router-level topology. While `traceroute`'s limitations are well-known [31] and commonly accepted, one of them may unduly impact our study. Specifically, `traceroute` reports a sequence of IP addresses that are matched against the responding router interfaces, and it is common for routers to have multiple interfaces. Multiple interfaces translate to multiple IP addresses belonging to the same router. Furthermore, *IP aliasing*[3] may generate even more IP addresses. In our study, working on an interface-level topology may lead us to overestimate the network size and coverage of network owners, or placing multiple computational units in the same router. As a consequence, an interface-level topology is not suitable for our aims. Fortunately, mapping network interfaces-level to router-level topology is a well-studied topic in networking, and we use CAIDA's IP aliasing resolution tool `kapar` [32]. To quantify the effect of IP aliasing, the size of the router-level topology is 50% smaller than interface-level topology (26K to 11K). We now describe how we generate hop- and latency-centric network graphs from the router-level topology.

1) Hop-centric network graph: Our data cleanup involves removing all routers which have private IP addresses (e.g., home routers) or are unresponsive (show up as $*$). Trimming the former is necessary since private IP addresses do not represent generic user-to-cloud connectivity, and their existence is largely dependent on how probes' owners configure their network. It must be noted that we aggregate all network latencies while removing private IP addresses to maintain the end-to-end latency estimate. Unresponsive nodes show up in our measurements due to in-network routers that disallow ICMP packets and, therefore, do not respond to `traceroute`s. Since we cannot determine neither ownership nor latency of such routers, we exclude them from our dataset.

While our cleanup techniques might result in a network graph with shorter paths, we believe that it represents a real Internet topology much closely. Moreover, since our analysis concerns the ownerships of network routers majorly, the trimmed network graph does not impact our results. Consequently, we generate three network graphs from our measurements, (i) *probe-to-cloud* (discussed in §III-A1), (ii)

[3]IP aliasing consists in associating more than one IP address to a single network interface

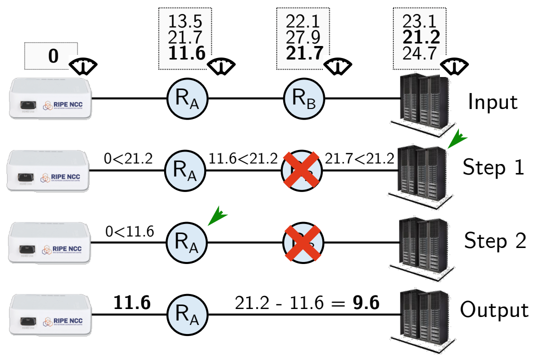


Fig. 2: Sanitizing RIPE Atlas `traceroutes` by removing nodes that violate monotonically increasing RTTs throughout the network path.

probe-to-probe (in §III-A2) and (iii) *probe-and-cloud* which unifies topologies in both (i) and (ii).

2) Latency-centric network graph: The objective of the latency-centric graph is to augment the hop-centric network graph with latency values per-hop. Due to the inconsistencies of latencies from `traceroutes`, our cleanup phase for generating a latency-centric graph is much stricter than the hop-centric graph. Every `traceroute` includes multiple measurements (typically three), which includes the RTT and the IP address of every hop. Since each measurement is independent of the last, there is a possibility that some routers earlier in the path yield higher RTT than routers that come later. This behavior is caused by different forward and reverse paths taken by ICMP packets for different measurements [33]. Let us consider the scenario illustrated in Fig. 2. The figure depicts the result of a `traceroute` issued by the Atlas probe towards the datacenter. Assume that the request identifies (through hop-centric graph processing) two routers on the path, R_A and R_B. The three RTT measurements recorded for each router are listed atop each node. Since RTT is the sum of base communication latency (limited by the speed of light) and delays due to traffic on the path, each of the three RTT values can vary significantly from another. We choose the minimum RTT (MinRTT) value reported for each router as base latency since our objective is to obtain the latency estimate of each hop least affected by additional network delays. We mark these values in **bold**. Using the MinRTT values, we can calculate the latency of each network hop by subtracting the MinRTT of the previous router from the one succeeding it, i.e., the latency of hop $R_A \leftrightarrow R_B$ is $MinRTT(R_B) - MinRTT(R_A)$. However, with non-monotonically increasing per-hop RTTs, such an approximation could result in negative latencies.

Our cleanup algorithm, shown in Fig. 2, works on the reverse path, i.e., from datacenter node to the probe. The first step of the algorithm starts from the destination node and proceeds backward until the source node. At every step, the active node (marked with a green arrow) ensures that all preceding nodes have MinRTT smaller than its own. In Step 1, we check that datacenter's MinRTT is greater than R_B, R_A, and the probe (which is assigned 0 RTT). In the considered example, R_B's MinRTT is greater than the datacenter, and therefore R_B is removed from the path. In the second step, the algorithm repeats itself, but from the perspective of R_A.

The trade-off for enforcing monotonically increasing RTTs is the reduced graph size, which does not resemble the hop-centric graph of the network. While our cleanup may reduce certain edge server deployment opportunities, it delivers, in turn, a consistent view of the network RTTs; this is quintessential for investigating the impact of edge computing on network latency. Similar to our hop-centric graph, we generate two latency-centric graphs: (i) *probe-to-cloud* and (ii) *probe-and-cloud* (we do not generate *probe-to-probe* since those paths do not culminate at cloud DCs).

IV. Measurements Analysis

In this section, we analyze the data collected from our measurements and investigate (i) the make-up of the underlying user-to-cloud connectivity over the Internet, (ii) the composition of shortest paths from probes to the nearest cloud, and (iii) the latency contribution w.r.t. the network ownership.

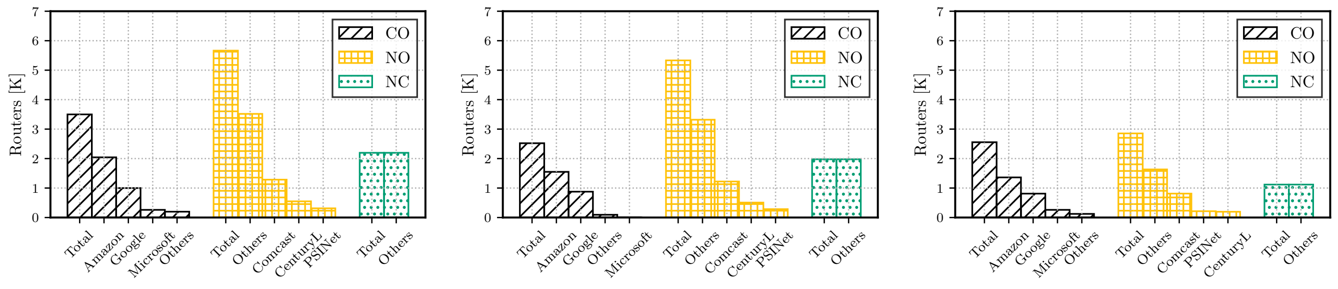
A. Owners Composition of the Network Graph

As discussed in § III, a typical user transits through several networks owned by different entities while connecting to a cloud datacenter. Understanding the entities that exist on such paths is critical for identifying potential players for edge server deployment and whether some of these players have an advantage over the others. We use our hop-centric graph for this scrutiny since it includes all the routers recorded throughout our `traceroute` measurements. The results of our analysis are shown in Fig. 3.

We find that, out ≈11K routers in *probe-and-cloud* network graph (shown in Fig. 3a), 30% belong to cloud operators (CO), 50% to network operators (NO), and the remaining 20% are unclassified (NC). From this, we infer that NOs own the majority of the in-network routers and, therefore, from a probability perspective, have much more edge server deployment space than COs. Interestingly, we find that Amazon, Google, and Microsoft collectively own the majority of the routers deployed by COs (95%). This is primarily due to the extensive datacenter deployment of the three operators, supported by their own private WAN infrastructure [34]–[36].

Fig. 3b shows the network composition of the *probe-to-probe* network graph (described in §III-A2). This graph includes ≈10K nodes – almost 86% of the entire hop-centric network. Such coverage from *probe-to-probe* measurements is somewhat expected since the majority of the path subsets (especially those operated by NOs) is covered by both mesh and cloud `traceroutes`. Surprisingly, even though our mesh measurements do not target cloud end-points, and we carefully remove probes located within DCs in our analysis, we find that $\approx 25\%$ of the routers in this graph belong to COs. While these may be routers leased by COs to NOs for directly peering user traffic to their private WANs [37], the result requires further investigation, which we leave for future work.

Fig. 3c shows the network composition of the *probe-to-cloud* network graph (described in §III-A1). The graph includes $\approx$ 8K IP addresses, of which 44% belong to NOs, 39%



(a) *probe-and-cloud*: 11358 routers (b) *probe-to-probe*: 9830 routers (c) *probe-to-cloud*: 6527 routers

Fig. 3: Ownership distribution over all the routers extracted from our `traceroute` measurements.

to COs, and 17% are unclassified (NC). It can be observed that the COs have a significantly higher router share in this graph compared to the *probe-and-cloud* graph. Thus, we deduce that many network segments present at the edge is not accessed when users connect to the cloud.

We now quantify the degree of the pervasiveness of COs in the whole network. To do this, we must isolate the network segments operated by COs in each network graph. The size of the *probe-and-cloud* graph (Fig. 3a), in terms of vertices, is denoted by S_W and can be decomposed into three main components. Being the union of the *probe-to-probe* and *probe-to-cloud* graphs, $S_W = S_E + S_C - S_{E \cap C}$, where S_E, S_C, $S_{E \cap C}$ are the vertices in the *probe-to-probe* graph (Fig. 3b), in the *probe-to-cloud* graph (Fig.3c), and in the overlapping vertices from both networks, respectively. Consequently, we can find $S_{E \cap C}$ as $S_E + S_C - S_W$. By substituting the values from Fig. 3, $S_{E \cap C}$ translates to 4999 routers. Hence, ≈44% of the whole graph is utilized for connecting towards 30 cloud DCs in the US. We further isolate the routers installed by COs at the network edge (Fig. 3b). Adapting the previous formula to assess the size of CO-only nodes, C, we calculate $C_{E \cap C} = C_E + C_C - C_W$ to be ≈1.5K nodes. Thus, more than 50% of the CO routers that are used to access the cloud DCs are also present in the *probe-to-probe* mesh measurements which *do not* even target cloud end-points. Therefore, we infer that CO-owned routers have already pervaded the network edge and are utilized to forward traffic that is not even destined to DCs. This phenomenon is also shown by a recent study about the flattening of the Internet [38].

Takeaways. The majority of network routers on user-to-cloud paths in the US belongs to network operators and ISPs (50%), making them preferred candidates for deploying edge servers. On the other hand, cloud operators are expanding their reach by installing routers within ISP networks, which directly peer traffic into their private WANs.

B. Owners Distribution on the Shortest Path to the Cloud

In this section, we investigate how the shortest path to the closest of the 30 DCs is partitioned among the three categories from §III-A. The overall aim is to quantify how much space is still available for edge servers deployment throughout the network paths and which of the categories have the most deployment potential. Before discussing the results, we want to emphasize that the plots do not refer to the raw `traceroute` result, but what is delivered after data processing from III-B and, therefore, paths will be shorter. Also, as we are interested in ownership, we ignore unresponsive nodes.

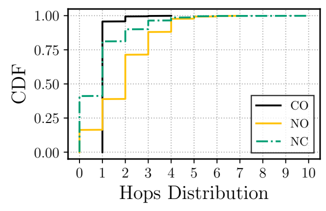
Fig. 4a shows the CDF of the number of hops (routers) belonging to each of the categories on the way to the shortest path to the closest DC. That is, if a path consists of 3 hops belonging to two NOs and one CO, three points will be mapped on the CDF, respectively NO=2, CO=1, and NC=0. From the figure, we see how COs are almost always present with one router. This result is to be expected as the end-point of every `traceroute` is indeed a CO. Also, a tiny fraction of paths have multiple, up to 4 hops, in a cloud network. Furthermore, roughly 20% of the shortest paths do not flow through any NO routers. This is because probes are one hop to the cloud, or due to unresponsive routers, or because the ISP belongs to the NC category. Moreover, in slightly less than 50% of the paths, there are no NC routers, and this reflects the accuracy of the classification of NOs. Many of the NC routers belong to small businesses and public institutions, e.g., universities with their own backbone.

Fig. 4b illustrates the hop distribution for the percentage of path belonging to each category. We see that COs consistently cover a significant part of the path that ranges from 20% to 50%. In rare cases, when the probe belongs to a CO but is not placed within the DC, COs covers up to 100% of the path.

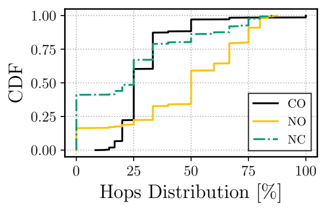
Takeaways. Cloud operators already cover a significant part of the paths to the cloud, and this reduces edge deployment space for NOs and NCs.

C. Latency Distribution on the Shortest Path to the Cloud

We now investigate how the shortest RTT towards the closest DCs is distributed among the three categories. Conversely to the previous section where we worked with IP addresses, we now consider latency on network edges, and this involves a source and destination IP. That is, we have to consider a couple of categories rather than individual ones. As a result, we will identify latency bottlenecks between inter/intra-category. The end goal is to identify where in the network edge servers deployment would benefit end-users. Here we use the latency-centric network graph, see III-B2.

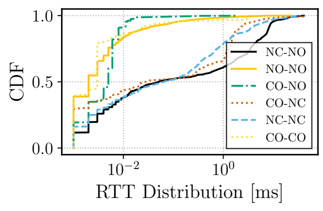


(a) Absolute number of hops

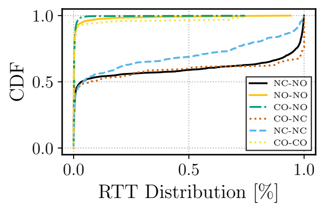


(b) Relative number of hops

Fig. 4: Ownership of the routers on the shortest path to the closest cloud DC for each probe.



(a) Absolute latency



(b) Relative latency

Fig. 5: Latency between every pair of routers on the shortest path to the closest cloud DC.

Fig. 5a depicts the absolute RTT distribution for the pairs of categories from §III-A. The classification in the legend has no order, meaning that CO-NO is the same as NO-CO. From the logarithmic scale on the x-axis, we can see that inter- and intra-category between NOs and COs is very efficient as it delivers almost in its totality sub-millisecond RTT. Hence, we see that the network segments between these two categories are really efficient. On the other hand, the remaining pair of combinations are not, and it is there that the bulk of latency lays. This suggests either congested links or poor links quality.

The percentage weight of these network segments within the shortest paths to the cloud is shown in Fig. 5b. Network segments between COs and NOs, as expected, contribute insignificantly to the overall shortest path latency. In some rare cases, few probes in the area of the closest DC obtain a high portion of the overall RTT. However, network segments between these categories contribute negligibly to the full RTT. From the CDF, we also notice that the bottleneck of the RTT is shared among the categories of the remaining network segments. That is, improvements to that portion of the network can dramatically reduce cloud access latency.

Takeaways. Network segments between the major network and cloud operators are very efficient. However, the latency bottleneck is shared among the remaining links.

V. Placement Strategies

In this section we present the edge placement strategies aimed to reduce end-users' access latency to the nearest compute server, but we first provide common notations. We define the set of candidate for edge deployment as $C = V \setminus \{P \cup E\}$, where P and E are the set of probes and cloud datacenters, respectively. This leaves us with every in-network router in the latency-centric network graph, except probes and DCs. The RTT matrix is $\mathbf{R}$, that is the RTT between every pair of nodes, and the the RTT between nodes i and j is obtained with $\mathbf{R}_{i,j}$. The selection function $\sigma(C, U, n)$ takes as input the set of candidates and, after the utility function U is applied to them, it returns the n candidates with the highest value. Intuitively, each placement strategy ranks candidates based on their utility function (e.g., RTT) that determines priority for edge deployment. That is, high utility nodes are preferred over those with lower value. Note that $\sigma(\cdot)$ is applied to every placement strategies in the evaluation phase. Through our results, we outline future research directions for maximizing the utility of edge on the Internet-wide scale.

A. Greedy

The greedy strategy is *eventually latency-optimal* as it delivers the best possible latency to all probes. This is achieved when every probe has an edge facility one hop away. The utility function for a probe $p \in P$ can be expressed as:

$$U(p) = \max \left\{ \frac{1}{\mathbf{R}_{p,\eta}} : \eta \in \mathbf{N}_p \right\} \tag{1}$$

where $\mathbf{N}_p \in C$ is the set of probe's neighbors. The rationale is that probes with highest latency gain are selected first. The drawbacks of greedy strategy are the high deployment cost and the inability to deliver a shared edge infrastructure, as users may be unwilling to grant access to others in the network.

B. Betweenness centrality (BC)

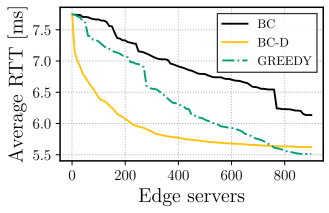
The goal of the BC strategy is to lower the end-users latency to the closest server without deploying as many servers as in the greedy strategy. *Betweenness centrality* (BC) indicates node centrality in a graph, and it is based on how many times a particular node is encountered through the shortest paths (Freeman's definition) [39]. BC has been employed widely, e.g., to maximize the reach of users in caching systems [40]. For edge deployment, betweenness centrality identifies aggregation nodes within a network, i.e., nodes mostly located on the shortest paths of other nodes. For example, nodes near the core of the network may be ideal locations to place edge and maximize reachability. The utility function of a candidate $c \in C$ is simply:

$$U(c) = \mathcal{B}_c \tag{2}$$

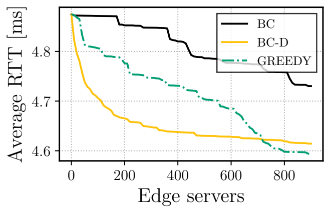
where $\mathcal{B}_c$ is c's BC. BC's complexity varies depending on implementations; we use Brandes' fast method that yields $O(VE + V(V + E) \log V)$ for weighted graphs [41].

C. Betweenness centrality with Depth (BC-D)

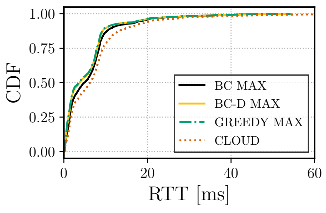
We believe that BC would be good to maximize reachability, however we hypothesize that the most central nodes in the graph (tree-like), see Fig. 1, are the ones closer to the cloud datacenters. If this is the case, candidate in-network routers for edge deployment would be gathered rather close to the datacenters. As a consequence, these nodes would be placed faraway from the end-users and will bring them little to no



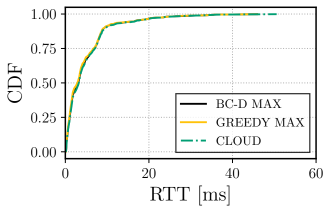
(a) *probe-to-cloud*: average RTT



(b) *probe-and-cloud*: average RTT



(c) *probe-to-cloud*: cloud vs. full deployment



(d) *probe-and-cloud*: cloud vs. full deployment

Fig. 6: Comparison between BC-D and GREEDY strategies over edge servers deployment. Note different y-axis values.

latency benefit. Our proposed BC-D algorithm solves this problem by weighting the BC's value by the RTT to the closest cloud DC. This way, nodes too close to the cloud, which would minimally reduce latency, are downgraded. Conversely, nodes further away from the cloud that may deliver better end-users latency will be preferred. Thus, we define the utility function of candidate $c \in C$ as:

$$U(c) = \mathcal{B}_c \cdot \mathbf{R}_{p,d^*} \tag{3}$$

where $\mathcal{B}_c$ is the BC of c, $\mathbf{R}_{p,d^*}$ is the RTT between the candidate node c and the closes cloud datacenter d^*.

VI. Simulation Results

We evaluated the three placement strategies from the latency point of view and how these target the different categories defined in §III. All the following results were obtained using the *probe-to-cloud* and the *probe-and-cloud* latency-centric graphs. We removed all probes that can reach the closest cloud DC in *one hop* as these do not have any room for edge deployment. After processing, *probe-to-cloud* and *probe-and-cloud* graph has 799 and 834 probes, respectively.

A. Edge computing latency benefits

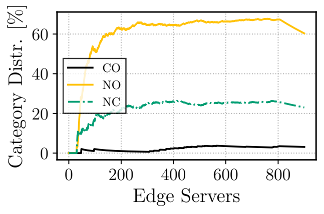
We start our analysis by calculating the access RTT for every probe. Access RTT is calculated on the shortest path toward the closest computational facility, e.g., DC or a newly placed edge server. If a probe has no edge server on the path, the access RTT equals the latency to access the nearest cloud.

Fig. 6a shows the evolution of the average access RTT for probes while increasing numbers of edge servers deployed in the graph. First, we see that the average RTT for accessing the closest cloud DC when no edge server is deployed is $7.8\,\mathrm{ms}$. On the other hand, when the number of edge servers is equivalent to the number of probes, the GREEDY strategy delivers $\approx$5.5 ms while the BC-D strategy $6.7\,\mathrm{ms}$. To put it differently, the GREEDY and BC-D strategies at peak edge deployment improves network latency by $\approx$40% and $\approx$33% respectively. More interestingly, we see that the two strategies perform equivalently when $\approx$ 750 servers are deployed ($\approx$95% of probes in the graph). Except for the remaining 25% of edge deployment, the BC-D strategy yields better performance. The constantly better performance of BC-D over the standard betweenness centrality, highlights the unsuitability of pure betweenness centrality in the context of edge placement.

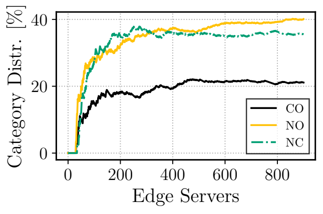
Fig. 6b shows the average access RTT for all probes, while an increasing number of edge servers is deployed in combined *probe-to-cloud* and *probe-to-probe* graph. In this graph, the average cloud access RTT (x-axis = 0) is $4.87\,\mathrm{ms}$. At peak edge deployment, the GREEDY strategy delivers an average access RTT of $\approx$4.6 ms while the BC-D strategy $\approx$4.65 ms, e.g., gain of $\approx$6% and $\approx$5%, respectively. Again, the two strategies' curves intersect, meaning that they are equivalent when the number of edge servers deployed is $\approx$750. Similar to our last result, BC-D delivers better access RTT than GREEDY until the edge deployment covers 95% of the total probes. Even for this graph, the standard BC is consistently worse than BC-D, showing poor performance in edge placement.

We now discuss the characteristics of the three strategies. We designed the BC-D strategy to decrease RTTs for multiple probes at the same time, as close as possible to the edge of the network; moreover, BC-D aimed to improve the efficiency of standard BC in the context of edge placement problems. The better performance, when compared to BC and GREEDY, clearly indicates that the design goals are met. Furthermore, as BC-D is also designed not to deploy too close to the probes, e.g., first hop, after a certain amount of deployments, it does not deliver further significant latency improvements, e.g., the curve flattens. On the other hand, GREEDY is supposed to optimize the RTT of one probe at the time, and this is reflected in the linearity of the curve of the average RTT's evolution over edge deployment. Latency-optimality is reached when edge servers cover all of the probes (on-premise deployment).

It must also be stressed that the efficacy of any deployment strategy is as good as the information of the network it is applied to. In this work, we augment our information of the network available to the user via mesh measurements, which allows for better deployment decisions. There are two major differences between the *probe-to-cloud* (Fig. 6a) and the *probe-and-cloud* (Fig. 6b) network graphs. Firstly, the *probe-and-cloud* (with edge paths) cloud access latency is $\approx$40% smaller than the *probe-to-cloud* (without edge paths) graph (value when no edge servers are deployed). This phenomenon strongly suggests that the paths being used to access the cloud are not optimal. Among possible causes, we believe the Border Gateway Protocol (BGP) routing to be the primary culprit. BGP is the Internet's default routing protocol, which decides the traffic forwarding decisions for routers at the crossings between ASes. The shortcomings of BGP are well-



(a) Greedy strategy



(b) BC-D strategy

Fig. 7: Percentage evolution of the three categories over edge servers deployment divided by strategy.

documented in the networking community [42]. The fact that BC-D algorithm, which is based on the shortest path to the closest cloud DC, delivers better results than default Internet makes us believe that there exist better routing opportunities on the Internet than what BGP offers. Currently, cloud traffic routing decisions are driven by peering agreements between providers operating at different portions of the Internet [37]. However, this matter requires further investigation left for future work. To sum up, our measurements suggest that cloud access latency in the US improves by ≈40% when non-conventional routes exploiting the network edge are used.

The second difference between the two graphs can be attributed to the impact of edge deployment on RTT reduction. The *probe-to-cloud* graph (without horizontal paths), after complete edge deployment, delivers an RTT improvement of 2 ms – reduction of ≈30% of the initial average RTT. On the other hand, the *probe-and-cloud* graph (with horizontal paths) reduces the initial average RTT by 0.27 ms, i.e. ≈6% improvement. We see two interesting renditions of these results. Firstly, potential edge deployment benefits *probe-to-cloud* graph more than *probe-and-cloud*. This suggests that the effectiveness of edge can be shadowed by sub-optimal network routing, and network operators should pay equal attention to both server placement strategies and cloud traffic pathing if they want to maximize the utility of their deployment. Secondly, from complete edge deployment with the GREEDY strategy (deployment achieves 1:1 mapping between probe and edge server), we observe that 85% of the average access RTT is concentrated in the last-mile link (probe to the first hop). Fig. 6c and Fig. 6d compare the cloud access latency with the full edge server deployment of both GREEDY, BC and BC-D placement strategies. From the figures, we can conclude that while it is possible to bring down latencies with aggressive edge deployment, it is non-trivial to accelerate performance for *all* users (exhibited by the long tails). Furthermore, Fig 6d shows that the impact of the edge in access latency reduction is even less when underlying networks are well-configured by their operators. However, since routing decisions can be influenced by performance-oblivious reasons, their end-to-end optimality cannot be guaranteed in the real-world. In such settings, effective edge deployment decisions are likely to bring immediate access latency benefits, which would increase with improvement in last-mile (wireless) access technologies.

B. *Owners of the network edge*

We now inspect which of the network owners (considered in Section IV-A) are most eligible for deploying edge servers using our placement strategies. Through our investigation, we provide insights on how edge computing could be brought to reality, more specifically, *by whom*. Fig. 7 shows the progression of the percentage of servers deployed by the different categories in the *probe-and-cloud* network graph (we leave out BC for relevance and space reason).

Fig. 7a presents the outcome for the GREEDY strategy that deploys servers as close as possible to each probe. Such a strategy directly impacts the server share of cloud operators (CO) – only 5%. On the other hand, the majority, ≈70%, of the deployed servers belong to the network operators (NO) as the first hop of the probe usually lies within the ISP infrastructure and provides the probe a point-of-entry to the Internet. The remaining ≈25% of the deployment is to be attributed to the non-classified (NC) category that is composed of university networks and minor NOs.

Fig. 7b shows the division of edge deployment among the three categories for BC-D. Recall that BC-D aims to lower the access latency as much as possible by sharing central nodes. From the figure, we see that the big chunk of these central nodes in the network still belongs to the NOs (40%) even though there is a decrease of ≈30% with respect to the GREEDY strategy. Conversely, COs increase their coverage by 15%, for a total amount of ≈20%. Finally, the non-classified (NC) nodes own the remaining ≈20% of the edge servers.

From the results, we clearly see that NOs, mainly providing Internet access, are in a dominant position, especially when compared to COs, for deploying a wide-scale edge. However, simply having the ability to place computational facilities closer to the users may not justify deployment. In fact, the results from Fig. 6b suggests that deploying edge closer to the users may not be worthwhile for the latency improvement that edge could bring.

VII. Discussion

In our study, we asked the question of whether edge computing could reduce cloud access latency for applications, and if yes, by how much. As a use case, we selected the US since it has the most extensive cloud data center coverage. Hence, it, in a way, represents the most difficult case for latency reduction because the clouds are already very widely spread. Our key finding is that while edge computing can indeed reduce cloud access latency, the reduction, in general, is very modest, on the order of 2 ms. While a reduction of a few ms may sound trivial, some applications, especially augmented reality, may depend on these, hence becoming viable with edge computing. Nevertheless, from a latency perspective, edge computing is far from a silver bullet in this case. Naturally, in regions with more sparse cloud deployment, the advantages are going to be more significant, and this merits a more extensive global study of cloud reachability. In our previous works [13], [14], we performed extensive global latency measurements to the cloud and their results indicate that in Europe and Oceania the latencies are comparable with the US, making it likely that our results would similarly apply in those regions. They also show that Asia, Latin America, and Africa have significantly higher latencies to the cloud, making them more appealing as deployment regions. However, these regions have only a small number of RIPE Atlas probes, making it difficult to obtain a good picture of the network topology. As future study, focusing on these regions would be of paramount importance.

Interestingly, our probe-to-probe measurements indicated the existence of shorter paths to the cloud than taken by the regular routing due to BGP and peering decisions between serving operators. We leave the investigation of these calls for further measurements. These also indicate interesting edge deployment potential, as it might be possible to get useful latency gains by going "sideways" to a neighboring network as opposed to going upwards to the cloud. This is something the network and cloud operators should look at together.

VIII. Conclusion

We focused our work on potential communication latency reductions that edge computing could bring to a country-wide network. We built such insights by collecting large scale measurements with the RIPE Atlas platform. We found that network operators, being majorly present in the network, are very good candidates for edge computing market domination. However, cloud providers already significantly pervaded into ISP networks, leaving poor space for deployment to network operators. Moreover, we conducted a thorough analysis aimed at identifying bottlenecks in the network, and we showed that cloud providers and network operators links exhibits good performance. Finally, we evaluated three placement strategies and estimated latency gains that hypothetical edge computing deployment could bring. Our finding suggests that either placing in-network servers or empowering network infrastructure, e.g., with peering agreements, could sensibly reduce network latency and enable a new class of latency-sensitive applications.

Acknowledgment

A special thanks to Suzan Bayhan that helped shaping the BC-D placement strategy in its early stage. This work is funded by SSF Future Factories in the Cloud (GMT-14-0032), Celtic project Piccolo (C2019/2-2), and Academy of Finland AIDA (317086) project.

References

[1] Y. Chen *et al.*, "An industrial robot system based on edge computing: An early experience," in *USENIX HotEdge 18*.
[2] M. S. Elbamby *et al.*, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.
[3] G. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.
[4] K. Ha *et al.*, "Just-in-time provisioning for cyber foraging," in *ACM MobiSys 2013*.
[5] W. Shi *et al.*, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.
[6] T. Taleb *et al.*, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, 2017.
[7] M. Satyanarayanan *et al.*, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, no. 4, pp. 14–23, 2009.
[8] Amazon, "CloudFront," https://aws.amazon.com/cloudfront/, 2020.
[9] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[10] RIPE NCC., "RIPE Atlas," "https://atlas.ripe.net/", 2020.
[11] L. Corneo *et al.*, "(how much) can edge computing change network latency?" 2021. [Online]. Available: https://mediatum.ub.tum.de/1609139
[12] A. Li *et al.*, "Cloudcmp: Comparing public cloud providers," in *ACM IMC 2010*.
[13] N. Mohan *et al.*, "Pruning edge research with latency shears," in *ACM HotNets 2020*.
[14] L. Corneo *et al.*, "Surrounded by the Clouds," in *The Web Conference 2021*, ser. WWW '21, 2021.
[15] P. Krishnan *et al.*, "The cache location problem," *IEEE/ACM transactions on networking*, vol. 8, no. 5, pp. 568–582, 2000.
[16] L. Qiu *et al.*, "On the placement of web server replicas," in *IEEE INFOCOM 2001*.
[17] B. Frank *et al.*, "Pushing cdn-isp collaboration to the limit," *ACM SIGCOMM CCR*, vol. 43, no. 3, pp. 34–44, 2013.
[18] I. Benkacem *et al.*, "Optimal vnfs placement in cdn slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 616–627, 2018.
[19] H. Wei *et al.*, "A new cache placement strategy for wireless internet of things," *Journal of Internet Technology*, vol. 20, no. 3, 2019.
[20] J. Liu *et al.*, "Cache placement in fog-rans: From centralized to distributed algorithms," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7039–7051, 2017.
[21] P. Radoslavov *et al.*, "Topology-informed internet replica placement," *Computer Communications*, vol. 25, no. 4, pp. 384–392, 2002.
[22] L. Wang *et al.*, "Service entity placement for social virtual reality applications in edge computing," in *IEEE INFOCOM 2018*.
[23] J. Xu *et al.*, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM 2018*.
[24] B. Gao *et al.*, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *INFOCOM 2019*.
[25] CAIDA, "CAIDA Archipalego (Ark) project," "https://www.caida.org/projects/ark/", 2020.
[26] H. V. Madhyastha *et al.*, "iplane: An information plane for distributed services," in *USENIX OSDI 2006*.
[27] A. Davis *et al.*, "Edgecomputing: Extending enterprise applications to the edge of the internet," in *ACM WWW 2004*.
[28] V. Bajpai *et al.*, "Lessons learned from using the ripe atlas platform for measurement research," *ACM SIGCOMM CCR*, vol. 45, no. 3, 2015.
[29] RIPE NCC., "Probe tags," "https://atlas.ripe.net/docs/probe-tags/", 2020.
[30] CloudHarmony, "Transparency for the cloud," "https://cloudharmony.com/", 2020.
[31] H. V. Madhyastha *et al.*, "A structural approach to latency prediction," in *ACM IMC 2006*.
[32] K. Keys, "Internet-scale ip alias resolution techniques," *ACM SIGCOMM CCR*, vol. 40, no. 1, 2010.
[33] E. Katz-Bassett *et al.*, "Reverse traceroute." in *NSDI*, 2010.
[34] A. W. Services, "AWS Global Infrastructure Map," "https://aws.amazon.com/about-aws/global-infrastructure/".
[35] Google, "Google Cloud," https://cloud.google.com, 2019.
[36] Microsoft, "Microsoft Azure," https://azure.microsoft.com/en-us/global-infrastructure/locations/, 2019.
[37] Google, "Google Direct Peering," "https://cloud.google.com/network-connectivity/docs/direct-peering", 2020.
[38] T. Arnold *et al.*, "Cloud provider connectivity in the flat internet," in *ACM IMC 2020*, 2020, p. 230–246.
[39] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977. [Online]. Available: http://www.jstor.org/stable/3033543
[40] L. Wang *et al.*, "Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking," in *ACM ICN 2015*.
[41] U. Brandes, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001. [Online]. Available: https://doi.org/10.1080/0022250X.2001.9990249
[42] T. Arnold *et al.*, "Beating BGP is Harder than we Thought," in *ACM HotNets 2019*.