UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2074

Networked Latency Sensitive Applications - Performance Issues between Cloud and Edge

LORENZO CORNEO

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2021

ISSN 1651-6214
ISBN 978-91-513-1295-8
URN urn:nbn:se:uu:diva-452971

Dissertation presented at Uppsala University to be publicly examined in Room 2446, ITC, Lägerhyddsvägen 2, Uppsala, Thursday, 4 November 2021 at 13:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Anna Brunström (Karlstads Universitet).

Abstract
Corneo, L. 2021. Networked Latency Sensitive Applications - Performance Issues between Cloud and Edge. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 2074. 62 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-1295-8.

The increasing demand for industrial automation has motivated the development of applications with strict latency requirements, namely, latency-sensitive applications. Such latency requirements can be satisfied by offloading computationally intensive tasks to powerful computing devices over a network at the cost of additional communication latency. Two major computing paradigms are considered for this: (i) cloud computing and (ii) edge computing. Cloud computing provides computation at remote datacenters, at the cost of longer communication latency. Edge computing aims at reducing communication latency by bringing computation closer to the users. This doctoral dissertation mainly investigates relevant issues regarding communication latency trade-offs between the aforementioned paradigms in the context of latency-sensitive applications.

This work advances the state of the art with three major contributions. First, we design a suite of scheduling algorithms which are performed on an edge device interposed between a co-located sensor network and remote applications running in cloud datacenters. These algorithms guarantee the fulfillment of latency-sensitive applications' requirements while maximizing the battery life of sensing devices. Second, we estimate under what conditions latency-sensitive applications can be executed in cloud environments. From a broader perspective, we quantify round-trip times needed to access cloud datacenters all around the world. From a narrower perspective, we collect latency measurements to cloud datacenters in metropolitan areas where over 70% of the world's population lives. This Internet-wide large-scale measurements campaign allows us to draw statistically relevant conclusions concerning the readiness of the cloud environments to host latency-sensitive applications. Finally, we devise a method to quantify latency improvements that hypothetical edge server deployments could bring to users within a network. This is achieved with a thorough analysis of round-trip times and paths characterization resulting in the design of novel edge server placement algorithms. We show trade-offs between number of edge servers deployed and latency improvements experienced by users.

This dissertation contributes to the understanding of the communication latency in terms of temporal and spacial distributions, its sources and implications on latency-sensitive applications.

Keywords: Latency Sensitive Applications, Cloud Computing, Edge Computing, Internet Measurements, Age of Information

Lorenzo Corneo, Department of Information Technology, Computer Systems, Box 337, Uppsala University, SE-75105 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-513-1295-8
URN urn:nbn:se:uu:diva-452971 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-452971)

To my parents, Laura and Tiziano, and my dear Heidi.

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I **Lorenzo Corneo** and Per Gunningberg. 2018. Scheduling at the Edge for Assisting Cloud Real-Time Systems. In *Proceedings of the 2018 Workshop on Theory and Practice for Integrated Cloud, Fog and Edge Computing Paradigms* (*TOPIC '18*). Association for Computing Machinery, New York, NY, USA, 9–14. DOI:https://doi.org/10.1145/3229774.3229777.

II **Lorenzo Corneo**, Christian Rohner and Per Gunningberg, Age of Information-Aware Scheduling for Timely and Scalable Internet of Things Applications, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2476-2484, DOI: 10.1109/INFOCOM.2019.8737497.

III Nitinder Mohan, **Lorenzo Corneo**, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks* (*HotNets '20*). Association for Computing Machinery, New York, NY, USA, 182–189. DOI:https://doi.org/10.1145/3422604.3425943.

IV **Lorenzo Corneo**, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. 2021. Surrounded by the Clouds: A Comprehensive Cloud Reachability Study. In *Proceedings of the Web Conference 2021* (*WWW '21*). Association for Computing Machinery, New York, NY, USA, 295–304. DOI:https://doi.org/10.1145/3442381.3449854

V **Lorenzo Corneo**, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. (How Much) Can Edge Computing Change Network Latency?. *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1-9, DOI: 10.23919/IFIPNetworking52078.2021.9472847

VI **Lorenzo Corneo***, Aleksandr Zavodovski*, Andreas Johnsson, Christian Rohner and Per Gunningberg. Analyzing Public Cloud Connectivity at its Best. *To be submitted.*
* co-primary authors.

Reprints were made with permission from the publishers.

List of papers not included

In addition to the papers that constitute this dissertation, I have authored or coauthored the following peer-reviewed papers. They are omitted from the thesis for the sake of a focused discussion.

- Mohammad Ashjaei, Kester Clegg, **Lorenzo Corneo**, Richard Hawkins, Omar Jaradat, Vincenzo Massimiliano Gulisano and Yiannis Nikolakopoulos. Service Level Agreements for Safe and Configurable Production Environments, 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), 2018, pp. 1252-1255, DOI: 10.1109/ETFA.2018.8502568.

- **Lorenzo Corneo** and Per Gunningberg. Towards Future Factories in the Cloud. 2018. In *14th Swedish National Computer Networking Workshop* (*SNCNW '18*), Karlskrona, Sweden.

- Victoria Catalán Rivas, Emil Fröjd, Tobias Holmberg, Felix Ragnarsson, Elsa Rick, **Lorenzo Corneo**, Ambuj Varshney, Christian Rohner, Thiemo Voigt, Per Gunningberg. Environmental Control at the Edge. 2018. In *14th Swedish National Computer Networking Workshop* (*SNCNW '18*), Karlskrona, Sweden.

- **Lorenzo Corneo**, Lukas Wirne, Christian Rohner and Per Gunningberg. The Grouping Window Schedule. 2019. In *15th Swedish National Computer Networking Workshop* (*SNCNW '19*), Luleå, Sweden.

- Walter Wong, **Lorenzo Corneo**, Aleksandr Zavodovski, Pengyuan Zhou, Nitinder Mohan and Jussi Kangasharju, Bricklayer: Resource Composition on the Spot Market, *ICC 2020 - 2020 IEEE International Conference on Communications*, 2020, pp. 1-7.
  DOI: 10.1109/ICC40277.2020.9149218.

- Ambuj Varshney and **Lorenzo Corneo**. 2020. Tunnel emitter: tunnel diode based low-power carrier emitters for backscatter tags. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (*MobiCom '20*). Association for Computing Machinery, New York, NY, USA, Article 42, 1–14.
  DOI:https://doi.org/10.1145/3372224.3419199

- Walter Wong, Aleksandr Zavodovski, **Lorenzo Corneo**, Nitinder Mohan and Jussi Kangasharju. SPA: Harnessing Availability in the AWS Spot Market. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2021, pp. 1-6, DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484646.

Contents

Acknowledgments

This doctoral dissertation is the result of almost five years of hard work supported by the great people that I met during my time as doctoral student. In the next paragraphs I will thank and acknowledge them.

First of all, I would like to thank my advisors at Uppsala University: Prof. Per Gunningberg and Prof. Christian Rohner. Per and Christian accepted me to be their student and allowed me to pursue my interests without any constraints. Being guided by them has been the best way to learn, and this made it much easier to achieve my goals. Per and Christian not only gave me freedom, but they also encouraged and supported me when I needed it the most. I will always remember when Per called me from the top of a mountain while skiing during his vacation just to give me feedback on our paper. I will also always remember having Christian remotely helping me during several all-nighters and during a road trip in Estonia (I was not driving). Thank you for all, Per and Christian.

I would like to thank all the colleagues at Uppsala University from CoRe and UNO as they made it incredibly fun and pleasant to get to the lab. Thank you Ambuj Varshney (for endless mentoring and long walks), Andreas Soleiman, Carlos Pérez-Penichet, Charalampos Orfanidis, Dilushi Piumwardane, George Daglaridis, Kasun Hewage, Laura Marie Feeney (a special thanks for helping with the language check of this dissertation), Sam Hylamia, Prof. Thiemo Voigt, Tobias Mages, Wenqing Yan. I am grateful for all the time spent together discussing about research, life and our dreams. This made the PhD journey simply memorable. I am also thankful to the teachers that I have been assisting during my time as TA: Prof. Lars-Åke Norden, Karl Marklund and Prof. Luca Mottola. It was a pleasure and a rewarding experience working for your courses. A special thanks to Andreas Johnsson for insightful discussions and for giving me the opportunity to work as a research intern at Ericsson Research. I am grateful to Prof. Mats Daniels, Alexandra Jimborean and Prof. Pontus Ekberg for following me in the senior group meetings. I want also to thank all the students that I had the pleasure to supervise and work with: Adnan, Ahmed, Elsa, Emil, Felix, Lukas, Robin, Tobias and Victoria.

I deeply thank Prof. Jussi Kangasharju for hosting me at the University of Helsinki and welcoming me in his research group. Jussi contributed significantly to my professional development and for this he became my co-advisor. I am honored and proud to have worked with him. I am also very thankful to all the Helsinki guys that made my stay in Finland really nice: Aleksandr Zavodovski, Nitinder Mohan, Otto Waltari, Pengyuan Zhou, Walter Wong. I thank you all for the nice brainstorming sessions, coffee breaks, and meetings

at Kaisla. I am also grateful to my co-authors that collaborated with me during the time spent in Helsinki: Prof. Suzan Bahyan (University of Twente) and Prof. Jörg Ott (Technical University of Munich).

I could have not reached the end of my studies without the help and support of my (big) family and friends. I am grateful to Filippo Guarnieri for mentoring and great discussions. Andrea, Annalisa and little Anita, thank you for the wonderful time spent together in Uppsala. I also thank Sirje, Marko, Ramona and the two furry superstars Domino and Caruso, for the stellar time we shared together. I warmly thank my dear parents, Laura and Tiziano, for always supporting and encouraging me to pursue my dreams, even when that meant a long distance between us. *Grazie 1000*! Finally, I thank my beloved Heidi, always there supporting and listening to me in tough times. Heidi made everything much easier and, most of all, funnier. *Aitäh, mu arm*!

Lorenzo Corneo
Uppsala, November 2021

This work has been funded by the Swedish Foundation for Strategic Research (SSF) under the project "*Future Factories in the Cloud*" (FiC) with grant number GMT14-0032.

List of Abbreviations

5G	5^{th} Generation
AoI	Age of Information
API	Application Programming Interface
AR	Augmented Reality
AS	Autonomous System
BC	Betweenness Centrality
BC-D	Betweenness Centrality with Depth
CDN	Content Delivery Network
CED	Cloud Edge Device
FZ	Feasibility Zone
GB	Giga Byte
HPL	Human Perceivable Latency
HRT	Human Reaction Time
ICMP	Internet Control Message Protocol
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISP	Internet Service Provider
LSA	Latency Sensitive Application
MEC	Mobile Edge Computing
MTP	Motion To Photon
MTU	Maximum Transmission Unit
PoP	Point of Presence
QoE	Quality of Experience
REST	REpresentational State Transfer
RTT	Round Trip Time
TCP	Transmission Control Protocol
TTL	Time Too Live
UDP	User Datagram Protocol
VR	Virtual Reality
WAN	Wide Area Network
WiFi	Wireless Fidelity

Part I:
Dissertation Summary

1. Introduction

Cloud computing has become essential for many networked services over the Internet [1]. Its ability to provide seemingly unlimited computational capabilities is one of the reasons for its world-wide adoption. Cloud computing provides optimized state-of-the-art hardware that is leased to the customers in virtualized environments, i.e., virtual machines. This way, cloud computing providers are able to maximize the usage of their underlying hardware infrastructure by sharing it among several users. Cloud computing provides a way to reduce complex computation times dramatically by elastically scaling up resources on-demand through convenient and flexible pricing models [2, 3]. For these reasons, cloud computing has become one of the key enablers of the Internet of Things (IoT): a large-scale network of connected devices via the Internet [4].

IoT applications usually involve sensing devices uploading sensory information to applications for external processing. Cloud computing providers support such applications by offering permanent storage and data analytic solutions for IoT generated data [5, 6]. Other services provide functionalities for triggering external applications or devices [7].

The vast amount of data generated by the IoT and the increasing demand for (industrial) automation have motivated the development of latency-sensitive applications (LSA). LSAs demand the fulfillment of one or more strict communication timing requirements. Common requirements are upper bounds on round-trip time (RTT) – the time for exchanging two messages between a sender and a receiver over a network – and RTT variation. Another timing requirement is upper bound on data freshness, which is a measurement of how old the information is [8, 9]. Examples of LSAs are augmented reality (AR), video streaming, and tactile Internet [10, 11, 12]. An application failing to satisfy timing requirements, such as RTT, leads to a degradation of the quality of experience (QoE) perceived by the user of the service, for example glitches in a video streaming application. In more severe cases, like in autonomous driving, missing LSAs requirements could lead to loss of lives and capital [13].

The growing need for applications with stricter latency requirements has led the research community to doubt the suitability of cloud computing as the main driver for LSAs. Datacenters used to be sparsely distributed around the world, and the propagation latencies induced by the geographical distance towards the users were too high to satisfy LSAs requirements [14]. Additionally, datacenters were solely accessed through the public Internet. Such shared infrastructure lacks the control over underlying resources, causing additional la-

tency, latency variation [15], and data staleness [8, 9], which are detrimental for LSAs.

These reasons motivated the research community to propose a new computing paradigm: edge computing [16, 17]. The core concept of this paradigm is to run the applications close to the users. This is done to avoid the variable communication latency introduced by the Internet and the highly virtualized environment of cloud datacenters. As a result, edge computing has become an attractive option for running LSAs.

However, during the last decade, cloud computing providers have constantly expanded their geographical coverage by extensively establishing new datacenters in different parts of the globe. This trend is set to continue in the years to come [18, 19]. As a result, the latency induced by the propagation delay to access cloud datacenters is getting smaller and smaller. Major cloud providers have been deploying private networks – with which they route network traffic destined to their datacenters – and established extensive peering agreements with major network operators [20]. This way, cloud providers are reducing the transit through the shared public Internet in favor of their privately managed infrastructure, thus improving datacenters' reachability and performance.

The advent of edge computing, coupled with the more established and always growing cloud paradigm, opens up new ways to deploy and distribute LSAs over the Internet. The main issue that needs to be tackled in the foreseeable future is to understand how to distribute LSAs between the edge of the network and cloud datacenters. The aforementioned exceptional technological advances in the fields of networking and computing are the key motivators for this doctoral dissertation. The main contribution of this work is to provide key insights for the deployment of LSAs over the Internet by better understanding the networking infrastructure between the IoT devices at the edge of the network and cloud datacenters.

1.1 Research Questions

IoT devices are one of the main drivers of LSAs since they deliver the data that LSAs process in order to build services. Such devices are usually powered by small batteries that would be depleted in a few hours of continuous operation. Thus, IoT devices prolong their battery life by sleeping most of the time and briefly resuming operation only for performing their duties, e.g., sensing and (wireless) transmission. IoT devices being non-operational most of the time, clearly conflict with the principles of LSAs that require the fulfillment of strict timing requirements, such as RTT and data freshness.

Edge computing gives new possibilities for application designers by simply placing an edge server between remote cloud applications and the sensor network, see Figure 1.1. The edge server can implement a cache hosting a digital replica of the sensor so that remote applications can access such infor-

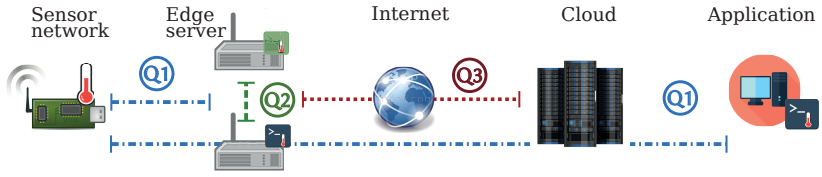


Figure 1.1. The circled signs $Q1$–$Q3$ locate the area of interest in the system model addressed by the research questions. Edge servers can prolong IoT devices' battery life while enforcing latency-sensitive applications' requirements, $Q1$. Edge computing offers alternative compute locations where to deploy applications, $Q2$, by having knowledge of the underlying networking infrastructure, $Q3$.

mation through a network. The edge server, through such cache, enhances the availability of the sensor network by always exposing the sensory information, masking the fact that sensing devices are on idle most of the time. However, the cache can not hide the fact that the cached sensory data becomes stale over time; certain LSAs can not tolerate stale information as it would lead the system to incorrect behaviors [8]. Given the intermittent nature of IoT devices and the strict requirements of LSAs, a compromise between data freshness, latency, and energy consumption must be achieved in order to guarantee the correct behavior of the applications. This motivates the first research question of this doctoral dissertation.

Q1 – How can edge computing support a sensor network to prolong its battery life while ensuring the fulfillment of LSAs' timing requirements?

A current issue with edge computing is where to place edge servers within the network. Early advocates envisioned a world of user-controlled mobile devices that opportunistically form processing pools for short-lived applications [21, 22]. Industrial standardization initiatives suggest edge infrastructure to be a component of the ISPs [23]. In the majority of the research literature, edge computing capabilities are implemented on commodity resource-limited devices, such as tablets or Raspberry Pis [16, 24]. Cloud providers implemented edge computing solutions, known as server-less computing, at existing content delivery networks (CDN) premises that have been expanded with computing capabilities [25, 26]. The diversity of the proposed solutions motivates further investigation to provide insights about the advantages that different edge computing deployment strategies could bring to LSAs. This compelling need for further understanding edge computing's benefits provides the context for the second research question.

Q2 – What benefits can an edge computing infrastructure provide to LSAs over the Internet?

It is possible to deploy an application at either a remote cloud datacenter or an edge server. Otherwise, applications can be deployed partly in the cloud, and partly at the edge. In order to devise a functional distribution of the parts of

networked applications, there is a need for understanding differences between execution in cloud datacenters and edge servers. This dissertation mainly investigates latency differences between these two options.

The proximity of edge servers to the users of LSAs would deliver low propagation delay. However, the edge computing infrastructure is not as mature and highly available as its cloud computing counterpart, that is globally available and easily accessible. Given the global availability and increasing pervasiveness of cloud computing infrastructures, there is a growing interest in understanding whether cloud datacenters could host LSAs. Such interest motivates the final research question addressed in this doctoral dissertation.

Q3 – Can LSAs that are deployed in cloud datacenters meet timing requirements, such as bounds on RTT and RTT variation? If yes, to what extent and in which regions of the world?

1.2 Contributions

This section describes the contributions of this dissertation that answer the research questions presented above, §1.1. The overall contribution of this doctoral dissertation is a better understanding of the networking infrastructure between the edge of the network and cloud datacenters, in the context of LSAs. Specifically, we answer the research questions $Q1$ by showing how to reduce energy consumption of IoT sensor networks with an edge server without compromising the timing requirements of LSAs. We answer the research question $Q2$ by devising an algorithm for placing edge servers within a network and assessing communication latency benefits over public cloud datacenters. Finally, we answer the research question $Q3$ by conducting Internet measurements towards public cloud datacenters to identify the limits of the cloud computing infrastructure. This way, we define the scope of edge computing to support LSAs. The contributions of this doctoral dissertation are summarized below in further detail.

Edge Computing Support for IoT Energy Constrained Devices
We find a solution to the problem of sensor networks' high energy consumption for delivering fresh sensory information to remote LSAs. We do this by designing a suite of scheduling algorithms that runs on an edge server co-located with a sensor network. The scheduling algorithm controls when device data should be collected and when it should be forwarded to the applications. The algorithm ensures that the latency requirements of the applications are met while minimizing the number of device readings for energy saving. The concept data freshness [8] of sensor data is used by the scheduler to ensure that the latency bounds are met (Paper I and II).

Edge Computing Servers Placement
We provide a method to estimate the latency experienced by the users after the deployment of edge servers in a network topology. We measure latencies from a source to a destination and from all visible routers on the path to the destination. This gives us a latency map that is combined with other source-destination pair measurements in order to devise a placement algorithm for edge servers. The algorithm minimizes the number of servers while still meeting the latency requirements of the considered applications (Paper V). The measurements data is publicly available at [27].

Cloud Reachability and Latency
The main contribution is a comprehensive latency measurement to public cloud datacenters all around the world from a large set of vantage points. Moreover, we thoroughly characterize network paths between the vantage points and the cloud datacenters. We provide fundamental knowledge for decision making concerning the deployment of LSAs over public cloud datacenters (Paper III and IV). From the comprehensive measurements we can identify "latency bottlenecks" links which is done with path analysis and detection algorithms [28] (Paper VI). The measurements data is publicly available at [29, 30].

1.3 Methods

The works included in this doctoral dissertation present different research methods that are used to evaluate algorithms and to characterize systems' behavior. The research methods include experimental measurements, trace-based and numerical simulations, as well as modeling. The advantages and disadvantages of each of them are discussed in the next paragraphs.

Experimental. Experimental measurements are used the most in this dissertation. The output of these measurements resulted in the collection of publicly available datasets that are fundamental building blocks of Paper III, IV, V and VI [27, 29, 30]. The strength of experimental measurements is the ability to capture real-world behaviors, such as the RTT between two hosts in a network. As a result, the collected measurements can approximate the behavior of the targeted parameters through a mathematical model. A consequence of experimental measurements is the difficulty to capture transient behaviors that may not repeat in time, e.g., latency measurement on a temporarily mis-configured network path. This aspect can be overcome only with long-lasting and frequent measurements, that in practice may be challenging to achieve.

Simulation. Simulation-based techniques allow reproducibility [31] of results in a deterministic way. Simulations are employed in Paper I, II and V. In Paper I and II, we use numerical simulations since the goal is to study in a deterministic manner the behavior of scheduling algorithms with respect to

LSAs requirements when perturbed by network latency. Hence, the focus is on boundary cases rather than realistic network behavior, that may not lead to capture the boundary cases we were interested in. In Paper V, we use trace-based simulations as we want to estimate the latency experienced by the users after the deployment of edge servers in a real-world network topology. We run Internet measurements and we use the collected dataset as input to the simulations. The trace-based simulations provide estimation of latency improvements without deploying real servers within the network.

Simulation-based techniques rely either on synthetic inputs, see Paper I and II, or on real-world traces like the ones collected in our measurements [27, 29, 30], see Paper V.

Modeling. Modeling techniques are useful when trying to decompose complex problems into simpler ones. In fact, a model abstracts away all the insignificant details and focuses on the core of the problem. In Paper I and II, we model the latency on a network link as deterministic, thus removing all the latency variations that a real network would deliver. This allows us to better understand the impact of deterministic latency and facilitates the understanding of the impact of latency variations. Models are extremely valuable to better understand the macro aspects of a system but they may not account for micro aspects, and this could lead to incorrect behaviors. For example, an algorithm designed on a model based on a deterministic network link may not work correctly on a link affected by latency variations.

2. Edge Support for Sensing Devices

This chapter discusses how edge computing can support real-time sensing devices that are also energy constrained. In particular, we provide background information on data freshness as a timing requirement for LSAs, and discuss cloud-edge-device (CED) systems. Edge computing can extend the battery life of sensing devices, while allowing LSAs to satisfy their requirements on data freshness and latency.

2.1 Data Freshness

The concept of data freshness was already a subject of study in the 1990s in database systems. In 1995 Adelberg et al. [32] raised the issue of keeping a database up to date while assuring time constraints on incoming transactions. In 2000, Cho et al. [33] elaborated and formalized the concept of data freshness as "age": the difference in time between the generation of the most recent information update and its use at a destination application. In more recent years, Kaul et al. [8] used the age metric in the context of vehicular networks for safety critical purposes. After that, the age metric has attracted significant attention from the research community and it is now commonly called Age of Information (AoI). The instantaneous AoI is defined as follows:

$$\Delta(t) = t - U(t) \tag{2.1}$$

where t is the current time and $U(t)$ is a function that returns the production time of the most recent update. From the definition follows that AoI grows linearly with time and drops to zero when a new update is received. The evolution over time of the AoI is shown in Figure 2.1. When AoI is used as timing requirement for LSAs, a maximum allowed value is usually defined. In Figure 2.1, the timing requirement on AoI, $\Delta(t)$, is defined as τ_Δ, and is represented as a horizontal dashed line. An application, in order not to violate such requirement, must have AoI less than the maximum allowed value, that is, $\Delta(t) \leq \tau_\Delta$.

Building on Equation 2.1, the concept of average AoI can be defined. Average AoI captures the long-term behavior of the AoI experienced by an application, and it is widely used in the literature. The average AoI over a period T is defined as:

$$\bar{\Delta}(t) = \frac{1}{T}\int_0^T \Delta(t)dt \tag{2.2}$$

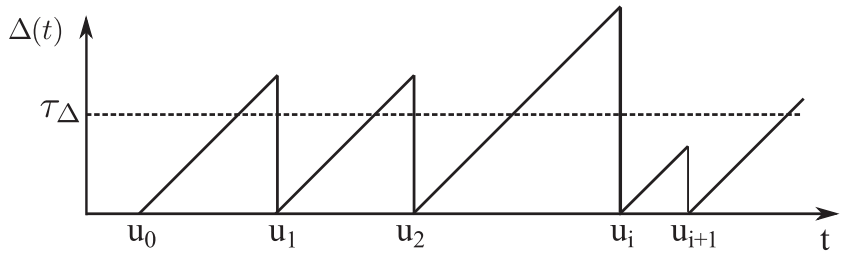


Figure 2.1. Evolution of the Age of Information, $\Delta(t)$, of sensory updates, u_i, over time. τ_Δ is the data freshness requirements over the sensory updates. That is, if the application consumes such information after the threshold τ_Δ, the system may function incorrectly.

Since the inception of AoI from Kaul et al., the properties of AoI have been studied in a variety of subjects and under very different scenarios. Instantaneous AoI (Equation 2.1) and average AoI (Equation 2.2) minimization problems have been studied in theoretical queuing systems in [9, 34, 35, 36, 37, 38]. Average AoI has been minimized with different approaches in the context of sensor networks under constrained sensor devices relying entirely on the energy harvested from the environment [39, 40, 41, 42, 43, 44]. In wireless communication, AoI-aware scheduling policies for a shared medium have been proposed in [45, 46, 47, 48, 49]. Optimal scheduling policies for systems with multiple information sources, e.g., multiple sensors, are provided in [50]. Average AoI minimization for energy constrained IoT sensing devices is studied in [51]. The optimality of the average AoI of updates gathering and dissemination within a graph with a random walker is addressed in [52].

It is also worth mentioning that the concept of AoI has evolved in new metrics that are variants of the original concept from [8]. For example, peak AoI introduced by Costa et al. registers the maximum AoI value experienced prior to the reception of a new update [36]. Also, with the concept of effective AoI, Yin et al. argue that not all the updates carry useful information, and hence they consider only the updates that are useful to the end-user for decision making [53].

2.2 Cloud-Edge-Device Systems

In this section, we discuss our vision of interoperability between the components of a Cloud-Edge-Device (CED) system, which is depicted in Figure 2.2. CED systems are composed of applications running in the cloud or remotely, one or more cloud datacenters, an edge server, and a sensor network. LSAs with requirements on data freshness subscribe to a cloud instance where they specify interest in particular sensors, and an upper-bound on AoI. An example of such an application is an industrial control system. Cloud virtual machines

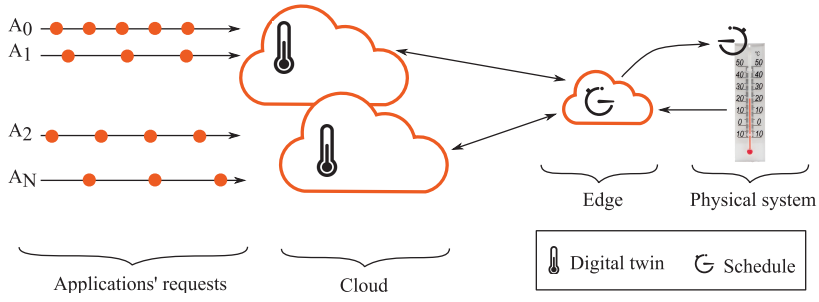


Figure 2.2. A Cloud-Edge-Device system – remote applications subscribe to sensor information with requirements on AoI. The edge device devises a schedule for the sensor network such that all the applications' requirements are satisfied, while minimizing energy consumption, e.g., the number of sensor updates.

export sensory information through "digital twins" [54]. A digital twin allows applications to access cached copies of sensor data as if they were accessed directly at physical sensing devices. However, because of the update rate of sensory information and the network communication latency, the digital twin exposes a digital copy of a sensor that has AoI greater than zero. When the AoI value is too high (not anymore fresh), applications such as control loops cannot function correctly and this situation should be avoided as much as possible. An edge server is located close to the sensor network and can receive subscriptions to sensors and data freshness requirements from the applications. Then, it aggregates all the requirements and combines them together, creating a schedule that can satisfy all the applications' requirements. The aim of this schedule is to instruct the sensors when to produce updates toward the edge server. Such a schedule can minimize the number of sensor updates transmissions, while providing the required level of data freshness. It can also account for communication delay towards the cloud infrastructure. A sensor network is typically made of small battery-powered sensors. Sensors are often duty-cycled in order to save energy and prolong their battery lifetime. Such duty-cycling is dictated by the schedule produced by the paired edge server. Once the edge server receives sensor updates, it will push them towards the respective digital twins in the cloud accessed by the applications.

In Paper I, we study how different scheduling policies affect the aging process of sensor updates on both edge servers and remote cloud datacenters. Paper I builds key insights for the problem formulation of Paper II. In Paper II, we propose and evaluate by simulations a suite of scheduling algorithms in an edge server for controlling the duty-cycling of a sensor network. Our aim is to minimize the energy consumption of the sensor network while still satisfying the AoI requirements. We achieve this by aggregating several LSAs' requests on the edge server. Our goal is in contrast to previous works that instead aim at minimizing the average AoI.

3. Internet Measurements

The Internet is a complex ecosystem composed of many independent components. This chapter briefly discusses the entities involved in the Internet, common diagnostic tools and platforms used to assess the state of the Internet. The information included in this chapter is essential background for the following chapters.

3.1 The Internet

The Internet is a network of networks, where every network comprises a combination of hosts and routers interconnected together. The following paragraphs provide a primer of the composition of Internet, common network protocols and diagnostic tools.

Autonomous Systems. A single network with a common prefix and a common routing policy in the Internet ecosystem is referred as an Autonomous System (AS). An AS is owned by an organization, e.g., Telia. For our purposes, we divide ASes into two groups, namely, edge ASes and core ASes; the former group provides Internet connectivity to hosts, while the latter group interconnects ASes. Edge ASes are typically Internet Service Providers (ISP) since they provide connectivity to the Internet to users. In the Swedish networking panorama, an example of a core AS is Telia, while Bredband2 and Comhem are edge ASes. Each AS is assigned an unique integer number by the Internet Assigned Numbers Authority (IANA), e.g., AS123. The number of allocated ASNs exceeded 100,000 as of July 2021 [55].

Internet Protocols. Hosts interact with the Internet by sending traffic using various protocols. The Internet Protocol (IP) provides routing functions that enable users to exchange packets across the Internet, hence achieving end-to-end reachability between them. Every networked device gets an IP address that identifies it within the network. There are two versions of IP: IPv4 and IPv6. Only IPv4 is considered in this dissertation. An IPv4 address is 32 bit long, and it is, for convenience, divided into 4 groups of 8 bits. The value of each of these 4 groups ranges from 0 to 255, in decimal. Thus, an IPv4 address could look like 172.46.58.1, where the 4 groups are separated by dots.

Other well-known and widely used protocols include the Internet Control Message Protocol (ICMP), the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). ICMP is a lightweight protocol which is intended only for control operations over the network, and is widely adopted by

network diagnostic tools. See next section. TCP and UDP are mainly used for sending and receiving application traffic.

3.2 Tools

This section presents two widely used network diagnostic tools, namely, `ping` and `traceroute`. These are both used in our measurements.

3.2.1 Ping

`ping` [56] is a tool that assesses the reachability between a source and a destination in the network using the ICMP protocol. In particular, `ping` issues ICMP Echo Request packets towards a destination that, if reachable, will reply to the source with ICMP Echo Reply packets. `ping` also provides an estimation of the RTT between them. `ping` is therefore extensively used to assess network latency between pairs of nodes in the network. It is worth mentioning that if the ICMP Echo Reply packet is not received at the source, it does not imply that there is no reachability between the nodes. In fact, some hosts or routers may be configured not to reply to ICMP Echo Requests and/or limit the rate of replies to such packets. The reason behind this is to avoid ICMP packets to interfere and possibly delay applications' traffic, which should be privileged.

3.2.2 Traceroute

`traceroute` [57] is used to reveal the path between a source and a destination in the network. The revealed path comprises the traversed routers between the source and the destination, and their IP addresses are listed by the tool. The path between the nodes is inferred by using the Time To Live (TTL) feature of IP packets, which is part of the IP header. TTL was introduced to solve the problem of IP packets looping on the Internet forever in case the destination IP address could not be found. The TTL is an integer that is decremented every time a router is traversed. When the TTL reaches zero, the packet is dropped and an ICMP Time Exceeded is returned to the sender. `traceroute` increments the TTL until a maximum value (30 as default) or until the destination is reached. This way, every router on the path returns the ICMP Time Exceeded packet and, by collecting their sender's addresses, it is possible to infer the full path between the source and the destination. `traceroute` also returns the RTT between the source and every IP address on the path towards the destination.

Limitations. `traceroute` presents some well-known limitations [58]. For example, it cannot account for load balancers along the path, an issue which

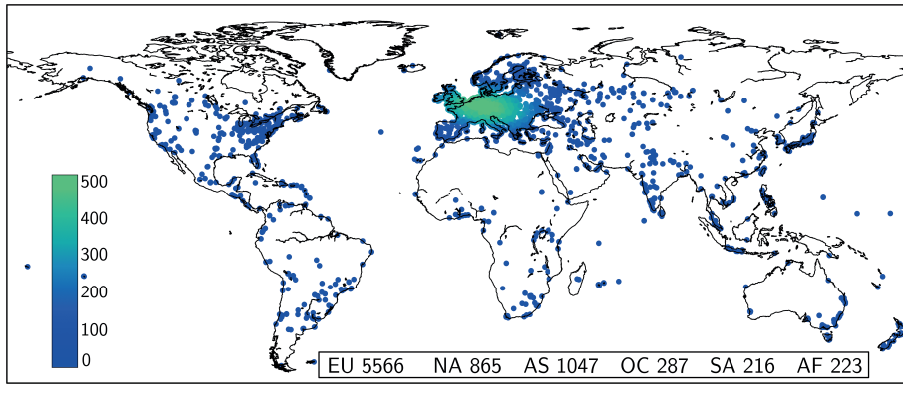


Figure 3.1. RIPE Atlas probes distribution of all the probes used to take all the measurements in this thesis. The color-bar indicates the density of Atlas probes installations in particular areas of the world, while the numbers in the box are the number of probes in each continent. The majority of probes are in Europe (EU), North America (NA) and Asia (AS). In our experiments, we used more than 5500 probes in Europe, 865 in North America, more than 1000 in Asia, and between 200 to 300 for the remaining continents (Oceania, OC, South America, SA, and Africa, AF). The probes are distributed in 184 countries all around the world.

is solved by Paris `traceroute` [59]. Also, `traceroute` reveals only the forward path, and completely neglects the return path. Not knowing the return path may limit the applicability of `traceroute` since most of the paths on the Internet are asymmetric [60] and have an impact on the RTT [61]. Information about the return path can be inferred with reverse `traceroute` [62]. Also, as previously mentioned, some nodes in the network do not respond to ICMP packets. Thus, the revealed path from `traceroute` may be only partially known.

3.3 RIPE Atlas

This section introduces the main features of the RIPE Atlas tool [63], which we used for all of our Internet measurements. RIPE Atlas is a globally distributed Internet measurements platform that is used extensively for reachability, connectivity, and performance studies. The platform hosts more than 12K vantage points, e.g., small hardware devices connected to the Internet via a wired connection. These hardware vantage points are also called *probes*, since they can emit probe measurement packets to desired destinations. Users can perform active network measurements (`ping`, `traceroute`, etc.) using RIPE Atlas vantage points towards end-points of their choice. RIPE Atlas probes are installed in heterogeneous network environments, such as core, access, and home networks, allowing users to analyze the reachability of networked devices globally. Atlas' vantage points are also hosted by cloud and network

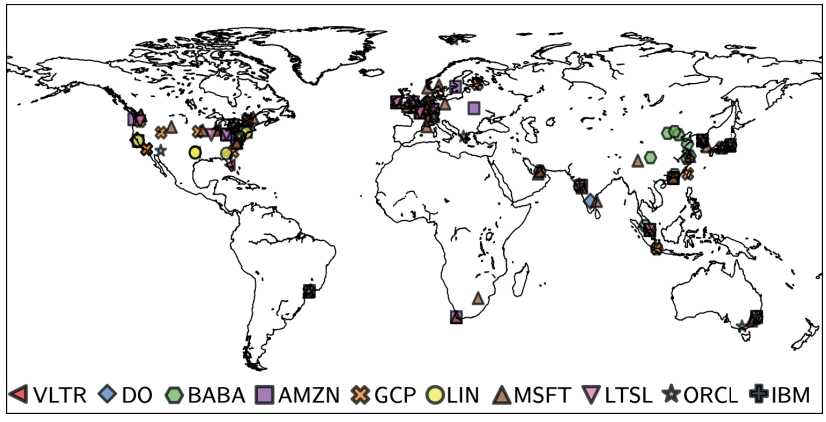


Figure 3.2. Approximate geographical location of all the datacenters used in the measurement studies part of this thesis. The current cloud infrastructure already serves North America, Europe, Asia and Oceania very well.

providers for monitoring their network reachability from outside their infrastructure [64]. Figure 3.1 shows the distribution of all the probes that we used in our network measurements. Most of them are in Europe, and roughly two thousands divided between North America and Asia. The remaining continents have between two and three hundreds of them.

RIPE Atlas offers a set of Representational State Transfer (REST) Application Programming Interfaces (API) [65] and a web application to interact with the platform. For every successful measurement scheduled, the platform generates a unique identifier that is used to download the results (fully or in batches) in the `JSON` format. Besides the APIs, RIPE also provides Python packages to design and start measurements, as well as to collect the results. For example, *Cousteau* is a Python wrapper that interacts with the RIPE APIs through an API key [66]. Moreover, *Magellan* is a Unix command-line interface to RIPE Atlas [67].

3.4 Cloud Computing Infrastructure

This section describes the cloud computing infrastructure used in the papers included in this dissertation. With a cloud computing infrastructure, we mean both the actual cloud datacenters, but also the private networks of the cloud providers. These networks support efficient communication between the datacenters of cloud providers and provide the end-users with paths that are faster than those offered by the public Internet. Global cloud computing services are provided by providers such as Amazon, Google, Microsoft Azure, IBM, Oracle, Alibaba, Digital Ocean, Linode, and Vultr. Their services are widely used, well-established, and provide global coverage with different networking

Table 3.1. *Breakout of the 189 datacenters that were used as targets in Paper III, IV, V, VI. We made use of 3 datacenters in Africa (AF), 58 in Asia (AS), 50 in Europe (EU), 62 in North America (NA), 4 in South America (SA) and 12 in Oceania (OC). The table specifies whether each provider has private, public of semi (private) network.*

		AF	AS	EU	NA	SA	OC	Network
Amazon EC2	AMZN	1	6	6	6	1	1	Private
Amazon Lightsail	LTSL	-	4	4	4	-	1	Private
Google	GCP	-	8	6	19	1	1	Private
Microsoft	MSFT	2	11	12	10	1	4	Private
Oracle	ORCL	-	7	4	4	1	2	Private
Digital Ocean	DO	-	1	4	6	-	-	Semi
IBM	IBM	-	1	6	6	-	-	Semi
Alibaba	BABA	-	16	2	2	-	1	Semi
Vultr	VLTR	-	1	4	9	-	1	Public
Linode	LIN	-	3	2	5	-	1	Public
Total: 189		**3**	**58**	**50**	**62**	**4**	**12**	

infrastructures. In fact, the networks interconnecting their datacenters are either private [18] or public, e.g., public Internet. It is also worth mentioning that some cloud providers sign agreements with major ISPs operating globally to enable direct peering between the ISP ASes and the cloud private networks point-of-presence (PoP) [68]. Some cloud providers even bypass Tier-1 network operators [20]. This is done to provide smaller communication latency to their customers.

In our latency measurements, we targeted 189 cloud datacenters in different regions of the world. Their geographical distribution is shown in Figure 3.2. As seen in the figure, most datacenters are in North America, Europe, Asia and Oceania. We specifically chose these public cloud providers since it is estimated that they collectively could serve over 70% of the world's population. The estimation is based on the fact that most of the population lives in metropolitan areas [69] and that datacenters in these areas will be in such proximity that the propagation delays will be negligible. This allows us to draw general conclusions about global cloud reachability. Table 3.1 lists the distribution of the targeted datacenters.

4. Edge Server Placement

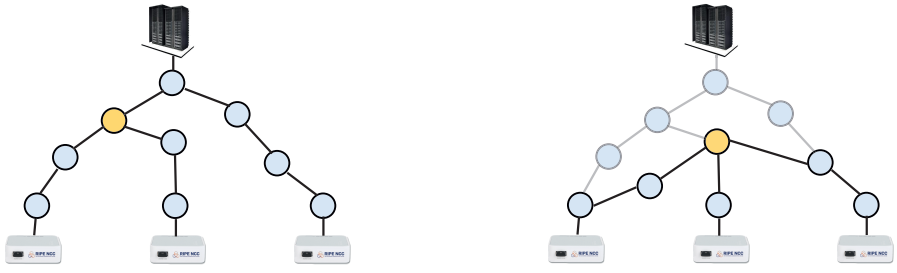
This chapter provides background information for the edge server placement problem addressed in Paper V. First, we list requirements for solving edge placement problems. We then provide related work around the CDN placement problem and discuss why the edge server placement problem is different. Finally, we consider the state of the art with respect to edge server placement, with particular attention to some relevant datasets and their limitations.

There are two fundamental requirements needed for solving edge placement problems, namely, (i) a network topology and (ii) placement strategies to evaluate. For requirement (i), both synthetic and real network topologies could be used. We use an inferred topology from our real-world measurements that account for real network latencies (see next section). The next paragraph briefly describes related placement strategies.

Placement problems have been addressed extensively in the context of CDNs. In the foundational work of Krishnan et al., the CDN placement problem is shown to be intractable in the general case and algorithmic solutions are given for several restricted variants [70]. Qiu et al. also investigate the issue in CDNs and suggest several algorithms, that are essentially approximations of the facility location or K-means problems [71]. Radoslavov et al. propose a method to harness network topology data for better CDN replica placement [72]. Frank et al. shows that the performance of CDN is enhanced by tightening cooperation with ISPs [73]. The CDNs placement solutions are usually based on cache placement strategies that try to balance the latency related to cache hits and cache misses.

Optimizing for cache hits is usually achieved by placement at central network locations, e.g., betweenness centrality, which however may involve a significant communication latency to the data source. Thus, the cache hit placement strategies are not suitable for edge placement problems, where the goal is to reduce communication latency when performing computation over a network. Also, latency based placement algorithms must include the probability that the latency requirements of the applications can not be met.

Placement problems are also addressed in edge computing research. For example, Wang et al. develop a combinatorial optimization algorithm focusing on service entity placement for VR applications [74]. For Mobile Edge Computing (MEC) environments, Xu et al. present an algorithm based on Lyapunov optimization and Gibbs sampling [75]. Gao et al. optimize placement in MEC environments by considering network performance [76]. However, these works concern the placement of software services, while we aim at placing hardware devices to support software deployment.



(a) Example network topology inferred by the *probe-to-cloud* measurements

(b) Example network topology inferred by the *probe-to-probe* measurements.

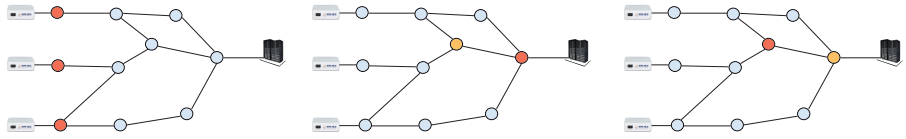
Figure 4.1. Examples of different network topologies discovered through our measurements from Paper V. The yellow node identifies the node with highest centrality and highest cumulative communication latency reduction.

In previous work, the authors attempt to map the Internet connectivity and routing at large [77, 78]. The datasets they use are not suitable for our edge server placement problem. For example, such datasets do not aim specifically at measuring latency between end-users and cloud datacenters, and hence cannot be used to evaluate placement strategies. Such limitation motivated us to collect a dataset of traces between RIPE Atlas vantage points and cloud datacenters in the USA, which we have made publicly available [27]. We use this dataset to evaluate placement strategies with the purpose to serve several users. These placement strategies trade off communication latency for the number of deployed servers. Our algorithm minimizes the number of deployed servers while still ensuring that the latency requirements are met by all users.

The following sections describe how the communication latency experienced by users is affected by different edge placement strategies.

4.1 Paths Discovery between the Edge and the Clouds

One key for good edge server placement is to have as many paths as possible between the users and cloud datacenters. This would deliver to the placement strategies a bigger solution space, and hence providing different alternative solutions that can be used for different objectives. The network path between the users and cloud datacenters is usually the result of peering agreements between network operators. It is possible to infer network paths using network diagnostic tools such as `traceroute`. Merging together network paths of several users results into a network topology that is a tree with the datacenter as root, and the users as leaves (see Fig. 4.1a) For convenience, we call this type of topology detection *probe-to-cloud* measurements. We argue that the topology obtained using *probe-to-cloud* measurements is too limited for our edge server placement problem.



(a) Greedy – edge servers are placed as close as possible to the probes.

(b) Betweenness Centrality (BC) – edge servers are placed at nodes with highest BC.

(c) BC with Depth (BC-D) – edge servers are placed at nodes with high BC and high latency towards the datacenter.

Figure 4.2. Different placement strategies used in Paper V. Red nodes are selected for edge deployment, orange nodes have been discarded in favor of red nodes.

A relevant factor for edge server placement is to identify nodes in the network that are traversed by the traffic of as many users as possible. These nodes are typically good *candidates* to host edge servers, since many users could reach and share them. Another important requirement is that these edge server *candidate* nodes should reduce as much as possible the latency for the users, when compared to cloud datacenters. This is possible only if these *candidate* nodes are "close", in terms of hops, to the users. *probe-to-cloud* measurements are successful in identifying network nodes shared by multiple users, but they may fail at identifying network nodes in proximity to the users. This is depicted in Figure 4.1a. The yellow node is shared by two users, but because it is close to the cloud datacenter, placing and edge server there does little to reduce the latency. It is common that these nodes are located close to (or within) the cloud operator's ingress, where the network traffic is aggregated before reaching the datacenter. We address this limitation by complementing the *probe-to-cloud* measurements with the *probe-to-probe* measurements, which are *mesh-based* `traceroute` measurements between users. The inferred network topology is used to identify additional nodes between the users that could host edge servers and further reduce the communication latency. Fig. 4.1b shows the additional paths discovered by *probe-to-probe* measurements. Note that the yellow node can be traversed by all the users and is closer to the users than the node identified in Figure 4.1a by the *probe-to-cloud* measurements.

Building on these insights, we use both measurement methods to collect real-world network paths towards cloud datacenters. We develop an edge server placement strategy that significantly reduces latency for users offloading computation over the Internet.

4.2 Placement strategies

In this section, we discuss the edge server placement strategies used in Paper V. We use simple visual examples from Figure 4.2 to better understand the differences between such placement strategies. Each placement strategy ranks network nodes as potential candidates for edge server placement, based on a

utility function, e.g., latency toward the closest datacenter. Such ranking determines the priority for edge deployment. Nodes with high rank are preferred over those with lower rank. Three placement strategies are described below.

Greedy. The greedy strategy is *eventually latency-optimal* as it delivers the best possible latency to all the users. This is achieved when every user can reach an edge server in one hop, see the red nodes in Figure 4.2a. The main drawback of the greedy strategy is the high deployment cost for the edge servers.

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 along the shortest paths (Freeman's definition) [79]. BC has been employed widely, e.g., to maximize the reach of users in caching systems [80]. 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 in the cloud operators' network may be ideal locations to deploy edge servers and maximize reachability, see Figure 4.2b.

Betweenness centrality with depth (BC-D). The BC strategy maximizes reachability with respect to the users. However, we hypothesize that the most central nodes in the graph are the ones closer to the cloud datacenters, see Fig. 4.1. If this is the case, network nodes that are candidates for edge deployment would be gathered rather close to the datacenters. As a result, these nodes would be placed faraway from the users and would deliver little benefit to reduce communication latency. If the BC strategy can not satisfy applications' latency requirements, the BC strategy can not be used to solve our edge placement problem. Thus, we devised the BC-D placement strategy. Our proposed BC-D algorithm solves the problem by multiplying the BC's value with the RTT to the closest cloud datacenter. This way, nodes too close to the cloud, which do little to reduce communication latency, are downgraded in ranking. Conversely, nodes further away from the cloud that may deliver better communication latency to the users will be preferred, see Figure 4.2c.

In Paper V we show that our BC-D strategy delivers communication latency reductions comparable to that of the greedy strategy, which delivers the best latency. However, our BC-D strategy demands only a fraction of the many servers deployed with the greedy strategy. We also show that the BC strategy does little to reduce latency for the users, because the central nodes in the topology are near the datacenters. As a result, the BC strategy is not suitable for edge server placements.

5. Cloud Reachability

This chapter describes the approach used in our Internet measurement studies to better understand the networking infrastructure between the edge of the network and cloud datacenters all around the world. We refer to this type of Internet measurements as *cloud reachability*, since they target cloud datacenters. The aim of our studies is to identify the limits of the global cloud infrastructure in terms of communication latency. The rest of the section describes significant related works in the area and how they differ from our cloud reachability studies.

The first significant cloud reachability study dates back to 2010 [14]. Li et al. present CloudCmp, a comparison of cloud computing platforms from the point of view of elastic computing, persistent storage, and networking services. Among several metrics, the RTT to access cloud datacenters is evaluated. More recently, the cloud performance report from ThousandEyes compares 96 cloud datacenters' network performance of major cloud providers – Amazon, Microsoft,Google, Alibaba, and IBM – for one month during the year 2019 [81]. The measurements methodology consists of collecting network latency and paths from 98 TCP-based vantage points located in Tier-2 and Tier-3 ISPs in 98 different locations around the world. Arnold et al. assess the flattening of the Internet by showing that several cloud providers bypass Tier-1 ISPs [20]. Thus, most of the users' traffic transits through the cloud providers managed networks, delivering better network performance to the users. Their methodology involves issuing ICMP `traceroute`s from virtual machines from six different cloud providers to a large number of IP addresses. Another work by Arnold et al. evaluates the effects of private WAN on cloud performance [82]. Palumbo et al. evaluate the latency performance of globally spread Amazon Web Services and Microsoft Azure datacenters from 25 vantage points from PlanetLab [83]. Haq et al. study inter-continental links between three major cloud providers and find them to deliver better RTTs and jitter when compared to public Internet links [84]. Tomanek et al. evaluate the latency performance of Microsoft Azure datacenters adopting a multidimensional approach [85]. Høiland-Jørgensen et al. study latency variation in the Internet from existing datasets of Internet measurements [15]. Agrawal et al. investigate the *unreachability* of cloud services and they find that depends not on the cloud network but on the last-mile links [86].

Paper III, IV and VI extend the state of the art described above by providing a broader perspective. Our measurements are far more comprehensive since they target 189 cloud datacenters and significantly more vantage points,

$\sim$ 8000, located in 184 different countries. This allows us to draw conclusions about cloud reachability from a global point of view. This was done in two ways. First, we measure cloud reachability from as many vantage points as possible for 1 year with a measurement frequency of 3 hours. Second, we measure cloud reachability for endpoints located in densely populated metropolitan areas where over 70% of the world's population lives [69]. We use a measurement interval of 15 minutes for at least 24 hours.

5.1 Latency Sensitive Applications

This section describes common latency-sensitive applications (LSAs) that we used to evaluate cloud reachability. We cluster applications based on their bandwidth utilization and latency requirements, quantified based on three well-known timing thresholds.

Motion To Photon. Motion To Photon (MTP) is the delay between the user input and its appearance on a display screen. MTP is key for immersive applications, such as AR/VR and 360° video streaming [87]. For these applications, a latency lower than 20 ms is important because the human vestibular system requires sensory inputs and interactions to be in complete sync. Failure to achieve such synchronization results in motion sickness and dizziness. Typically, $\sim$ 13 ms are taken by the display technology to refresh its content and this leaves a budget of $\sim$ 7 ms for communication delays and computation on local or remote server [88].

Human Perceivable Latency. The Human Perceivable Latency (HPL) threshold is reached if the delay between the user input and the visual feedback becomes long enough to be detected by the human eye [89]. The HPL threshold plays a key role in the quality of experience (QoE) of applications where the user interaction with the system is fully or semi-passive, e.g., video streaming buffering, cloud gaming input lags, etc. The HPL is estimated to be $\sim$ 100 ms.

Human Reaction Time. The Human Reaction Time (HRT) is the delay between the presentation of a stimulus and its associated motor response by a human. While the HRT depends heavily on the individual and can be improved by training, its value is reported to be $\sim$ 250 ms [90]. Latencies for applications that require active human engagement, such as remote surgery and tele-operated vehicles, must be within the HRT bounds.

5.1.1 Classification

Considering the similarities in operational thresholds, we now group emerging LSAs with respect to their latency and bandwidth requirements (Figure 5.1a).

Quadrant I - *Low Latency & Low Bandwidth*: The bottom-left green quadrant represents applications that produce only a small volume of data but impose

strict latency constraints for correct operation. Typical examples include wearables, health monitoring, and other human-focused applications. The core aim of applications in *Quadrant 1* is to perform "naturally", i.e., to operate within the HPL threshold.

Quadrant II - *Low Latency & High Bandwidth*: The bottom-right blue quadrant encompasses applications that generate large data volumes *and* impose strict latency constraints. Typical examples include autonomous vehicles, AR/VR and cloud gaming. Edge computing is considered a key enabler for applications in *Quadrant 2*. In fact, edge computing reduces communication latency and bandwidth bottlenecks towards cloud datacenters, hence avoiding possible disruption of the immersiveness of the end-user [91].

Quadrant III - *High Latency & High Bandwidth*: The top-right yellow quadrant is composed of applications that generate large volumes of data but have relaxed latency constraints. For example, a smart city provides automatic updates on bus timetables and smart parking meters.

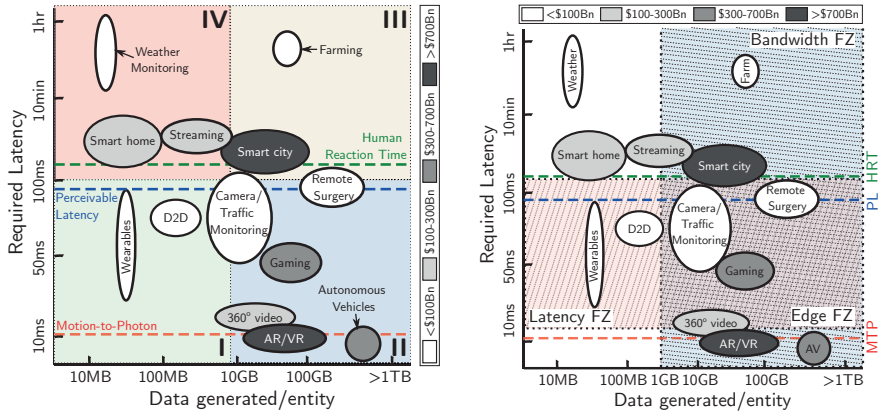
Quadrant IV - *High Latency & Low Bandwidth*: The top-left red quadrant consists of applications that neither generate data of large volumes nor require strict latency for correct operation. Typical examples include smart homes and weather monitoring.

5.2 Edge Computing Feasibility Zone

This section discusses the approach taken in Paper III to identify LSAs that benefit from the low communication latency provided by edge computing solutions. We measured latency between users and cloud datacenters. Knowing the communication latency between users and cloud datacenters allows us to identify LSAs that can not run in the cloud because of the violation of latency constraints. These LSAs may need edge computing solutions.

For the measurements, we chose 101 cloud datacenters around the world from seven cloud providers, namely, Amazon, Google, Microsoft Azure, Digital Ocean, Linode, Alibaba, and Vultr. This is a subset of datacenters listed in Table 3.1 and shown in Figure 3.2. We used over 3200 RIPE Atlas probes [63] distributed in 166 countries as vantage points for our measurements. A superset of these vantage points is shown in Figure 3.1. We measured the end-to-end latencies between RIPE Atlas vantage points and the above cloud datacenters within the same continent via `ping` every three hours. For probes in continents with low datacenter density, i.e., Africa and South America, we also measured latencies to datacenters in adjacent continents, i.e., Europe and North America. Overall, the dataset includes 3.2 million data points, and it is available for public usage [29].

Figure 5.1a shows the LSAs and their network requirements. Figure 5.1b is identical to Figure 5.1a, but adds the results from our measurements. The red



(a) *Latency-sensitive applications represented as ellipses. The orientation and width signify strictness in bandwidth and latency requirements. The gray scale colors denote the expected market share by the year 2025 [92].*

(b) *Latency-sensitive applications with edge computing feasibility zone. The red shaded area represents communication latency from our measurements, and the blue shaded region is the bandwidth gain zone for utilizing edge.*

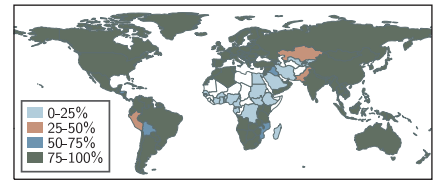
Figure 5.1. Placement of common latency-sensitive applications into quadrants where the dimensions are latency and bandwidth requirements.

shaded region shows the range of our latency measurements to cloud datacenters. We see from these measurements that some datacenters are accessed in as low 10 ms latency, and that most datacenters can be reached within 250 ms.

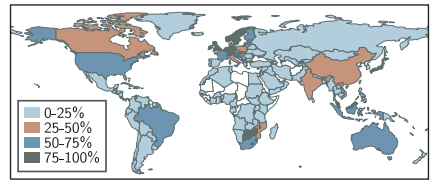
Our latency measurements are not designed for bandwidth estimation. Thus, based on existing literature, we define a minimum bandwidth that may motivate the usage of an edge server. For bandwidth, edge computing is mostly useful for applications generating so much data that leads to network or datacenter congestion. As a result, the network benefits the most when an edge server is placed as close as possible to the application that is generating large volumes of data. It is also well-known that the primary source of bandwidth bottleneck is usually the so called *last-mile*, which is the network segment between the user and the ISP [93]. Previous study indicates that applications generating at least 1GB per second of data can benefit from edge computing [94]. This area is depicted in Figure 5.1b as a blue rectangle.

In Figure 5.1b, we define the overlapping area between the red and the blue regions as the "*feasibility zone*" (FZ) of edge computing, see *Quadrant 2*. Applications within the FZ, such as traffic camera monitoring and cloud gaming, could benefit the most from the deployment of edge servers as they impose both latency and bandwidth constraints. Surprisingly, important use cases used to motivate edge computing deployment *do not* fall in the FZ. For some, it is due to low bandwidth requirements, e.g., wearables, and for others, it is either too stringent or too relaxed latency constraints, like autonomous vehicles and smart cities, respectively. Interestingly, the predicted market share of applica-

(a) *Vantage points RTTs within the Human Reaction Time (HRT),* $\leq$ 250 ms.



(b) *Vantage points RTTs within the Human Perceivable Latency (HPL),* $\leq$ 100 ms.



(c) *Vantage Points RTTs within the Motion-to-Photon (MTP),* $\leq$ 20 ms.

Figure 5.2. Global coverage of measurements from Paper IV with respect to the three timing thresholds defined in §5.1

tions [92] within the edge FZ is much less than those for which edge computing does not provide much benefit. In conclusion, many applications on the edge FZ can be supported by a wider deployment of cloud datacenters and network infrastructure.

5.3 Latency Sensitive Applications in the Clouds

This section describes the methodology used to assess the possibility to run LSAs in cloud datacenters around the world. The following discussion is based on the work conducted in Paper IV and Paper VI.

5.3.1 General Cloud Reachability

For the measurements, we chose the 189 datacenters listed in Table 3.1. As vantage points, we selected over 8000 probes from the RIPE Atlas [63] distributed in 184 countries across the globe. Most of the selected probes were located in Europe and North America, 33.5% and 26.5%, respectively. Our main objective was to analyze the *user-to-cloud* latency at global scale, hence we measured end-to-end latencies between users and the cloud datacenters with `ping`. We configured the RIPE Atlas vantage points to `ping` all of the available datacenters (Table 3.1) on the same continent every 3 hours throughout the measurement period. For vantage points in continents with a low datacenter density, i.e., Africa and South America, we also included `ping` RTTs to datacenters in adjacent continents, i.e., Europe and North America, respec-

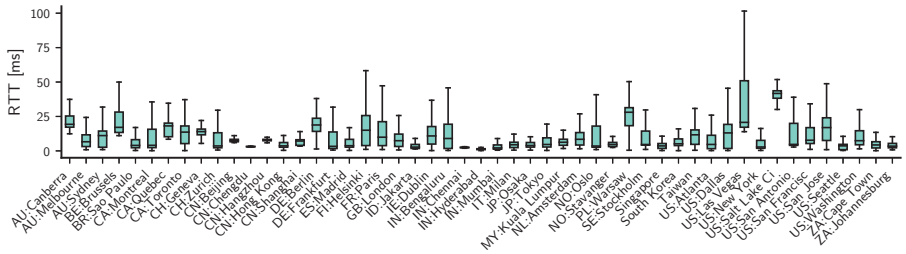


Figure 5.3. RTT distribution of RIPE Atlas vantage points located in metropolitan areas with population bigger than 1M and at least one datacenter nearby ($\leq$ 50 km).

tively. We quantified the end-to-end distance, as hop count, between users and cloud datacenters via `paris-traceroute` [59] measurements. Besides the ICMP-based `traceroutes`, we also launched `TCP traceroute` and recorded the per-hop RTTs. Unlike the ICMP based measurements, which may not reach the final destination because of firewalls and/or ICMP rate limiting, our TCP measurements are end-to-end. This provides an accurate representation of latencies encountered by real applications operating in the cloud, e.g., a web server. More than 4M unique paths were identified: 6880 in Africa, 8345 in South America, over 450K in the United States of America, over 630K in Asia, and nearly 3M in Europe. This represents more than 60 GB of measurement data. The whole data set is freely available at [30].

5.3.2 Cloud Reachability in Metropolitan Areas

In the previous section, we highlighted our approach to general cloud reachability at global scale, involving as many vantage points as possible. In this section, we discuss our methodology to measure cloud access latency from vantage points installed in 51 metropolitan areas, each with a population bigger than 1M, and having at least one datacenter in their proximity (max. 50 km). The rationale behind this choice is that, according to United Nations Population Division [69], over half of the global population now lives in cities. Especially high is the percentage of the urban population in Northern America (83.6%), Southern America (81.2%), and Europe (74.9%). We selected over 700 RIPE Atlas [63] vantage points and 114 cloud datacenters from Amazon, Google, Microsoft, IBM, Alibaba, Oracle, DigitalOcean, Linode, UpCloud and Vultr.

In our measurement campaign, we used `paris-traceroute` [59] to measure RTT from a vantage point to a cloud endpoint. We launch probes with an interval of 4 minutes to each nearby cloud endpoint for a period of at least 24 hours. We used the TCP variant of `paris-traceroute`, which uses TCP `SYN` probing packets instead of ICMP, and avoids low-prioritization or blocking of ICMP packets. We used full-sized MTU packets of 1500 bytes to mimic the packet size of real applications.

5.4 Discussion of Results

We conclude this chapter by showing the high level results from our measurement studies. In particular, we show the extent to which the global cloud infrastructure can satisfy the network latency requirement of LSAs. Figure 5.2 illustrates the global RTT distribution from all the vantage points in our measurements, one for each timing threshold, as defined in §5.1. Different color groups denote different percentiles of the distribution. The results suggest that almost every country across the globe can consistently reach the closest cloud datacenter within the boundaries of the HRT, 250 ms. In fact, only 2 (out of 184) countries in our dataset achieve HRT less than 25% of the times, and 3 countries lie between 50 and 75%. For HPL, 100 ms, we observe that the situation changes slightly when compared to the HRT. In fact, 140 countries achieve RTTs consistently within the boundaries of the HPL, 6 achieve it only 50 to 75% of the times, another 6 within 25 to 50% and, 16 countries meet the HPL threshold less than 25% of the times. The countries experiencing degraded RTT are mainly clustered in Central Africa, the Middle East, and South America. The distribution changes substantially for the MTP threshold, 20 ms, where only 24 countries consistently meet it between 75 to 100% of the times. Conversely, 125 countries meet the MTP threshold less than 25%, and the remaining 25 countries can reach it between 25 to 75% of the times.

Figure 5.3 shows the RTT distribution of the measurements from the vantage points in 51 metropolitan areas with a population bigger than 1M and with at least one datacenter nearby ($\leq$ 50 km). The results show that most of the vantage points in the selected metropolitan areas can already meet even the MTP latency. This result is important as it suggests that for the most of the world's population [69], the network does not prevent LSAs with strict requirements from being successfully executed in cloud datacenters.

From our results, we deduce that the networking infrastructure between users and cloud datacenters does not hinder networked applications from meeting HRT and HPL latency. This opens up scenarios in which LSAs could run in cloud datacenters in most regions of the world. LSAs requiring MTP latency pose strict constraints concerning the distance between applications and datacenters. In fact, we found that cloud datacenters located less than 50 km away from metropolitan areas can be reached by nearby users with average latency compatible with MTP. However, latency variations caused by shared infrastructures, such as the Internet and datacenters, may still prevent correct execution of certain LSAs.

6. Summary of Papers

Paper I

Lorenzo Corneo and Per Gunningberg. 2018. Scheduling at the Edge for Assisting Cloud Real-Time Systems. In *Proceedings of the 2018 Workshop on Theory and Practice for Integrated Cloud, Fog and Edge Computing Paradigms* (*TOPIC '18*). Association for Computing Machinery, New York, NY, USA, 9–14. DOI:https://doi.org/10.1145/3229774.3229777.

Summary

We study edge server support for multiple periodic real-time applications located in different clouds datacenters [95]. The edge server communicates both with sensor devices over wireless sensor networks and with applications over the Internet. The edge server caches sensor data and can respond to multiple applications with different latency requirements to the data. The purpose of caching is to reduce the number of multiple direct accesses to the sensor, since sensor communication is very energy expensive. However, the data will then age in the cache and eventually become stale for some application. A push update method and the concept of Age of Information [8] are used to schedule data updates to the applications. An aging model for periodic updates is derived. We propose that the scheduling should consider periodic sensor updates, the differences in the periodic application updates, the aging in the cache and communication latency variations. By numerical analysis, we study the number of deadline misses for two different scheduling policies.

Reflections

In this paper, we build the foundation for the work conducted in Paper II. Here we started the study of how latency is distributed between sensor networks, edge servers and cloud datacenters. By using Age of Information as a requirement for latency- sensitive applications, we characterize how information produced by the sensor network ages in edge servers and in cloud datacenters, when the network is perturbed with additional delay variations. This work is relevant to system designers of latency-sensitive applications that demand the fulfillment of data freshness requirements.

My Contributions

I am the main author of this paper. I conducted all the technical work with the supervision of Prof. Per Gunningberg. The manuscript was written by both me and Prof. Per Gunningberg, and I mainly focused on the description of scheduling policies and evaluation.

Paper II

Lorenzo Corneo, Christian Rohner and Per Gunningberg, Age of Information-Aware Scheduling for Timely and Scalable Internet of Things Applications, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2476-2484, DOI: 10.1109/INFOCOM.2019.8737497.

Summary

We study large scale Internet of Things applications requesting data from physical devices. Specifically, we address the problem of timely dissemination of sensor updates towards applications with freshness requirements through a cache. We aim to minimize direct access to the possibly battery powered physical devices, yet improving Age of Information [8] as a freshness metric.

We propose Age of Information aware scheduling policies that run at an edge server in proximity of a sensor network. The aim of the scheduling policies running on the edge server is to support the physical devices to push sensor updates to caches in cloud data centers [96]. The freshness requirements vary from applications to application and the scheduling policies exploit this to group together applications' requests. By delaying the transmission of sensor updates, it is possible to satisfy several applications' requests with a single sensor update without violating applications' deadlines. As a result, the sensor network reduces the number of sensor updates sent to the cloud. Also, the scheduling policies ensure that the applications' deadlines can be met given a specified delay variation of the network. The policies are incrementally introduced as we study them, first, over a deterministic communication link and, second, over a link with unpredictable delays according to a statistical model. We numerically evaluate the proposed scheduling policies against a simple yet widely used periodic schedule. We show that our informed schedules outperform the widely used periodic scheduling policy, even under high delay variation.

Reflections

The scheduling policies presented in this work reduce the energy consumption of a sensor network without sacrificing the AoI experienced by the remote

applications. We show that the widely used periodic scheduling policy does not deliver satisfactory data freshness, despite it is rather energy efficient. We devised scheduling policies that outperform the periodic schedule, and are efficient both in terms of energy consumption and data freshness. This work shows a novel application for edge computing and is a reference example of cooperative interaction between edge servers and cloud datacenters.

My Contributions

I am the main author of this paper and I wrote the most of the manuscript. I devised the idea of scheduling at the edge for improving AoI of sensory updates for cloud-based applications. I devised the base scheduling algorithm and refined its variations with the help of Prof. Christian Rohner.

Paper III

Nitinder Mohan, **Lorenzo Corneo**, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks* (*HotNets '20*). Association for Computing Machinery, New York, NY, USA, 182–189. DOI:https://doi.org/10.1145/3422604.3425943.

Summary

In this paper, we scrutinize edge computing research, examining its outlook and future considering recent technological advances [97]. We perform extensive client-to-cloud measurements using over 3000 RIPE Atlas vantage points scattered in 166 countries. We ran ICMP `ping` every three hours for 4 months between September 2019 and December 2019 towards 101 datacenters in 21 countries around the world.

We show that latency reduction as a motivation for edge computing is not as strong as perceived. We found that in well-connected areas, like Europe, North America and Oceania, the cloud infrastructure can satisfy almost all application requirements that have been envisioned for edge computing. The remaining continents may remain infeasible for immediately foreseeable future, as they depend on still developing networking infrastructure or on last-mile wireless access latency. The cellular technology 5G promises to improve the performance of the last-mile but global roll-out will take several years to complete. Recent studies over 5G also show sub-optimal results [98]. Our results showed that, from a performance point of view, the potential benefits of edge computing remain small. Africa, Latin America and parts of Asia may benefit from edge computing where the density of cloud centers is relatively low and/or the

networks are less advanced. Therefore, we believe that the research in edge computing should balance the latency-centric view with studies on edge services for privacy, data aggregation and others.

Reflections

The contributions in this work are intended both to stimulate discussions within the networking community regarding the future of edge computing research, and to assess to which extent the current cloud infrastructure can run latency-sensitive applications. This paper was appreciated by the research community as it clarifies where and when edge computing solutions are required to run applications with specific latency requirements.

My Contributions

The paper was lead by Nitinder Mohan. I developed an expertise in using the RIPE Atlas platform and collected the Internet measurements; I also analyzed the results. I contributed to the writing of the measurements methodology and results sections.

Paper IV

Lorenzo Corneo, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. 2021. Surrounded by the Clouds: A Comprehensive Cloud Reachability Study. In *Proceedings of the Web Conference 2021* (*WWW '21*). Association for Computing Machinery, New York, NY, USA, 295–304.
DOI:https://doi.org/10.1145/3442381.3449854

Summary

We present a comprehensive cloud reachability study motivated by the continuous expansion of cloud networks and the lack of large-scale recent assessment [99]. We perform extensive global users-to-cloud Internet latency measurements towards 189 datacenters from 9 major cloud providers. We run ICMP `ping` as well as ICMP and TCP `traceroute` between vantage points and all the datacenters within a continent. From the measurements, we evaluated the suitability of modern cloud environments for latency-sensitive applications by comparing our latency measurements against known human perception thresholds. Thus, we could draw inferences on the suitability of current cloud infrastructures for latency-sensitive applications. Our results show that the current cloud coverage can support several latency-critical applications for

most of the world's population. We highlight that the performance of private cloud networks is slightly better, or at least comparable to the performance obtained on the public Internet.

Reflections

This work presents an overview of the connectivity to cloud datacenters, from almost anywhere in the world. This research assesses which latency-sensitive applications can be run in cloud datacenters, and which need alternative solutions such as edge computing support or similar local processing technologies. Cloud operators could benefit from this study as we identify where additional cloud infrastructure should be deployed in order to further reduce communication latencies towards datacenters.

My Contributions

I am the lead author of this paper and I wrote most of the manuscript. I did all the `ping` measurements, and I analyzed the measurements results. I also devised the concept of *cloud pervasiveness*. Maximilian Eder provided preprocessed data for the `traceroute` measurements that he also collected.

Paper V

Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. (How Much) Can Edge Computing Change Network Latency?. *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1-9,
DOI: 10.23919/IFIPNetworking52078.2021.9472847

Summary

The core of the edge computing paradigm is to bring the computation from a distant cloud datacenters closer to service consumers and data producers. Consequentially, the issue of edge computing facilities' placement arises. In this paper, 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. We present a comprehensive analysis suggesting where to place general-purpose edge computing resources on a discovered network topology [100]. We base our conclusions on extensive large-scale Internet `traceroute` measurements to cloud datacenters in the USA, leveraging the RIPE Atlas platform [63]. We identified the affiliations of the routers to determine if network providers can act as edge providers. We

devised edge placement strategies. From our measurements on our discovered network topology, we estimate that our devised placement strategies can reduce latency up to 30% with edge servers. However, the absolute values of the reductions remain on the order of few milliseconds 2 ms to 3 ms, and may not justify the deployment of edge computing facilities in countries with extensive cloud datacenters deployment, such as the USA.

Reflections

This paper aims to solve the real-world problem of estimating communication latency reductions by additional deployment of computing facilities, e.g., edge servers. The adopted methodology is broad and complex, since it involves competence in Internet measurements, graph processing and algorithm design. I believe that the information included in the paper could be useful to cloud and network operators for expanding their networks and computing facilities. The proposed methodology generally applies to any network topology.

My Contributions

I am the lead author of this paper and I wrote most of the manuscript. I did all the measurements and performed data analysis. I implemented all the software to perform graph processing, and I devised the placement algorithms, with the help of Nitinder Mohan, Prof. Suzan Bayhan and Prof. Christian Rohner. The research direction has been identified together with Prof. Jussi Kangasharju.

Paper VI

Lorenzo Corneo, Aleksandr Zavodovski, Andreas Johnsson, Christian Rohner and Per Gunningberg. Analyzing Public Cloud Connectivity at its Best. *To be submitted.*

Summary

Cloud computing has a remarkable growth not only in terms of its capacity and performance but also in its geographical coverage and density, coming closer to the users. With shorter network distances to users, it is gradually overcoming its initial drawback – the high latency of access. Thus, we are interested in quantifying the network RTT in well-connected and largely populated metropolitan areas from regular user points on the Internet. Besides RTT, we also measure its variation since it is of paramount importance for latency-sensitive applications. In addition, we identify bottleneck links in the network

paths that are responsible for most of the RTT variations, e.g., by packet queuing, and hence poor performance. This study covers over 50 metropolitan areas around the world and over 100 public cloud datacenters. The RIPE Atlas platform is used for the measurements. The collected measurements exhibit the median RTT to the nearest datacenter of 6.49 ms with a standard deviation below 1.23 ms, for 50% of the samples. We also find that cloud ingress is often responsible for the deterioration of performance.

Reflections

This work quantifies the communication latency towards datacenters that users living in metropolitan areas experience. This work is motivated by the fact that over 70% of the world's population lives in metropolitan areas. This work is then to be considered a benchmark that can be used as a reference for researchers and application designers when distributing applications over the Internet. Cloud operators could use our study to inspect and improve their network locations where we detected high RTT variations.

My Contributions

I am the co-primary author of this paper and I wrote most of the measurement parts of the manuscript together with Aleksandr Zavodovski, which conducted the Internet measurements. Together, we analyzed the gathered measurements. The research direction was devised with Dr. Andreas Johnsson.

7. Future Work

In this chapter, I will present what I believe are promising research directions that have as the starting point the topics addressed in this dissertation. In particular, I will discuss some limitations experienced in our Internet measurements campaign, suggestions on how to overcome them, and the future of LSAs.

Cloud computing and wireless technologies. Paper III, IV, V and VI assess Internet connectivity to cloud datacenters from vantage points wired to the Internet. However, extraordinary advances in the field of miniaturization and ultra low-power radio technologies suggest that the future of computing devices is more and more wireless oriented. These trends open up future research targeted at a better understanding of the impact of different wireless technologies with respect to cloud datacenters connectivity over the Internet. Measurement platforms like Speedchecker [101] count hundreds of thousands of wireless vantage points. Speedchecker can be used to extend our global cloud reachability evaluation, as well as to give insights about latency contributions of wireless technologies. Speedchecker is still a relatively new platform and there is a need to understand its advantages and disadvantages to better understand how it complements RIPE Atlas.

Another promising wireless technology that deserves deep investigation is 5G. A number of 5G latency evaluation studies have been presented in recent years [98, 102]. However, no broad cloud reachability study has been conducted with respect to Internet access through 5G networks. 5G is an appealing topic for research since it has different configurations that affect the end-to-end latency, e.g. URLLC, and no study addressing this aspect has been presented yet. The big challenge ahead of a broad evaluation of cloud datacenters access via 5G networks is to provide globally distributed vantage points. At the moment of writing, there is no measurement platform providing such service, and this suggests that there is a big opportunity for developing one that aims at performance study of 5G devices over the Internet.

Knowing the impact of wireless technologies would help to provide a better understanding of whether LSAs could be deployed in cloud datacenters, while being accessed (or supervised) through wireless networks.

High frequency and broader network measurements. The RIPE Atlas platform restricts the amount of simultaneous measurements and their frequency in order not cause congestion in the network or impact the results. This impacts the observability of the networks, especially on transient events that may not be captured. While such restrictions are put in place for security and fairness

reasons, I believe there is a need for a measurement platform that, by design, allows high frequency measurements to reveal insights that cannot be captured with RIPE Atlas and similar platforms.

RIPE Atlas provides a small subset of tools, such as `ping` and `traceroute`. Performing available bandwidth measurements is hard, if not impossible, to do with RIPE Atlas. I argue that a specialized platform for highly frequent measurements should be devised in order to capture transient events that can help to better understand the Internet. An example of such platform could be a distributed measurements system like RIPE Atlas or Speedchecker, but with the diversity of measurements provided by M-LAB [103].

Application layer measurements. The measurements in this doctoral dissertation only account for the communication delay – sum of propagation, transmission and queuing delays – and do not account for applications' processing time, which is essential to estimate end-to-end latency. This is challenging to estimate because one of the biggest questions for edge computing research is centered on the computational capabilities of edge servers. Specifically, there is not yet a consensus on the computational capacity that edge servers will offer to the users, and no direct comparison with the hardware already deployed in cloud datacenters. Therefore, a promising research direction is to investigate the processing delay of applications running on different hardware to help system designers in better understanding and estimating end-to-end latency for LSAs that offload computation to either edge or cloud servers.

The future of latency-sensitive applications. The insights from this dissertation and the aforementioned future works are just means to understand the distribution of the latency in the many entities involved in networked LSAs. Recent trends in the field of networking and computing show that the Internet and the cloud/edge computing infrastructures will be mature enough to deliver latency compliant with the motion to photon. As a result, latency will no longer be a hindrance to deploying LSAs remotely, either in cloud datacenters or in edge servers. However, addressing latency variations will still be a prominent issue because of the shared infrastructures supporting LSAs.

I believe that the biggest challenge will be the orchestration of LSAs over the Internet. The big questions to be answered will be *where to deploy functionalities of LSAs, in particular where to deploy functionalities that handle sensitive data?*

8. Conclusion

The fast-paced technological advances of the past years have dictated a growing demand for reliable latency-sensitive applications. Given the increasing and global wide deployment of datacenters, the cloud computing paradigm has become very attractive to run latency-sensitive applications. However, the highly virtualized nature of cloud datacenters and their access through the Internet pose questions regarding the ability to satisfactorily meet the stringent latency requirements of remote applications. The edge computing paradigm emerged with the objective to bring computation closer to the applications, hence avoiding communication latency on the Internet between users and datacenters. This doctoral dissertation investigates the networking infrastructure between the edge of the network and cloud datacenters, with the ultimate goal to demarcate the latency domain of the two computing paradigms, cloud and edge computing, with respect to latency-sensitive applications. The following paragraphs provide answers to the research questions stated in § 1.1

Research question $Q1$ asked *how edge computing could support IoT devices in reducing their energy consumption, while still meeting latency-sensitive applications' requirements*. We envisioned edge servers interposed between sensor devices, sensor access networks, and remote cloud datacenters in order to schedule timely sensor updates for remote applications. In this context, edge servers coordinate and mediate between sensor networks and remote applications via cloud datacenters. We conclude from our research that edge servers can be used to minimize IoT devices' energy consumption, and at the same time provide timely and up-to-date content to the applications.

Research question $Q2$ asked *which latency benefits an edge computing infrastructure can bring to latency-sensitive application over the Internet*. We answered this question by providing a measurement methodology that estimates latency gains between the edge of the network and cloud datacenters. We validated our methodolgy by doing measurements with the RIPE Atlas platform towards cloud datacenters in the USA. Our results suggest that edge servers deployment could cause a gain of 2 ms to 3 ms over direct access to cloud datacenters. We believe that the main reasons for this limited improvement are the wide deployment of cloud datacenters in densely populated areas, as well as an efficient networking infrastructure in these areas. Therefore, we conclude that edge computing will be most beneficial in areas with lower density of cloud datacenters and/or with less efficient networking infrastructures.

Research question $Q3$ asked whether *cloud datacenters could satisfy the requirements of latency-sensitive applications*. We answered this question by

conducting extensive Internet measurements towards cloud datacenters all around the world and assessing the latency performance of the underlying networking infrastructure. Our results show that a RTT of 100 ms to cloud datacenters is achievable almost everywhere on planet Earth. Continents hosting a vast deployment of datacenters and metropolitan areas with nearby datacenter(s) can deliver RTT to datacenters below 20 ms with some tolerance for RTT variations. Such RTT would already allow several latency-sensitive applications to run on cloud datacenters.

Svensk Sammanfattning

Nya tekniska framsteg har gett högre prestanda inom moln- och Internetteknologier vilket öppnar upp för att kunna köra nya klasser av real-tidsapplikationer på molnet över Internet. Det är fördelaktigt att köra beräkningstunga applikationer i molnet, på virtuella maskiner, för att avlasta lokala datorer och utnyttja molnets skalbarhet. Augmented reality är ett exempel på en sådan applikation. Realtidsapplikationer ställer höga krav på fördröjningar och andra tidsbegränsningar på nätet för att fungera korrekt.

Två paradigmer är aktuella för att stödja applikationer som är känsliga för fördröjningar: (i) etablerade molntjänster som erbjuder realtidsexekvering (ii) och det senare "edge computing" (sv. en server som är strategiskt placerad i nätet, närmare användaren än molntjänsten). Dessa två har olika egenskaper. Denna avhandling studerar vilka tidskritiska applikationer som passar endera paradigmen och kombinationer där emellan utifrån latensmätningar på Internet till verkliga molntjänster.

Avhandlingen presenterar tre vetenskapliga bidrag. Det första är en svit av schemaläggninsalgoritmer som körs på en edge server strategiskt placerad mellan användarens utrustning och molnet. Algoritmerna schemalägger när data skall hämtas från utrustningen och när data skall levereras till molnet för att klara de tidsbränsningar som applikationen i molnet har. Ofta består utrustningen av trådlösa sensorer som drivs av batterier. I det sammanhanget är det viktigt att inte i onöden belasta sensorerna med läsningar som drar ur batteriet. Algoritmerna reducerar antalet läsningar till ett minimum samtidigt som de ser till att leverera sensor-data till applikationerna enligt specifierade gränser.

Det andra bidraget är latensmätningar i Internet till öppna molntjänster för att förstå vilka klasser av applikationer som kan köras på molnet. Mätningarna är gjorda från ett stort antal punkter i Internet till den närmaste publika molntjänsten. Dessa mätningar är gjorda under lång tid från mätpunkter spridda över hela världen. I ett smalare perspektiv har vi gjort mer omfattande mätningar i metropoler som svarar för mer än 80 procent av världen befolkning. Dessa omfattande och globalt täckande mätningar gör att vi kan dra statistiskt relevanta slutsatser i vilken omfattning molntjänster idag kan hantera tidskritiska applikationer.

I det tredje bidraget utvecklar och utvärderar vi en metod för optimal placering av edge servers i nätet för exekvering av real-tidsapplikationer jämfört med i molnet. Här utgår vi från latensmätningar som mäter varje länk i nätet till molnet. Routers mellan länkarna betraktas som möjliga placeringar av edge

servers. Vår nydanande placeringsalgoritm analyserar olika placeringsalternativen utifrån den trade-off som finns mellan antalet edge servers som placeras och den potentiella minskningen i fördröjningen i nätet.

De vetenskapliga uppsatserna i denna avhandling bidrar till en djupare "state-of-the art" kunnade om kommunikationsfördröjningar för distribuerade realtidsapplikationer över Internet. Målgruppen för avhandlingen är systemdesigners av tidskritiska applikationer.

Bibliography

[1] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13):2009, 2009.

[2] Robert L Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[3] Rahul Singh, Prateek Sharma, David Irwin, Prashant Shenoy, and KK Ramakrishnan. Here today, gone tomorrow: Exploiting transient servers in datacenters. *IEEE Internet Computing*, 18(4):22–29, 2014.

[4] Dave Evans. How the next evolution of the internet is changing everything. 2011.

[5] IBM. Watson iot platform. `https://www.ibm.com/cloud/watson-iot-platform`, 2021.

[6] Inc. Amazon. Aws iot. `https://aws.amazon.com/iot/`, 2021.

[7] Inc. IFTTT. Ifttt. `https://ifttt.com/`, 2021.

[8] S. Kaul, M. Gruteser, V. Rai, and J. Kenney. Minimizing age of information in vehicular networks. In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 350–358, June 2011.

[9] S. Kaul, R. Yates, and M. Gruteser. Real-time status: How often should one update? In *2012 Proceedings IEEE INFOCOM*, pages 2731–2735, March 2012.

[10] Wenxiao Zhang, Bo Han, and Pan Hui. On the networking challenges of mobile augmented reality. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, VR/AR Network '17, page 24–29, New York, NY, USA, 2017. Association for Computing Machinery.

[11] Robert Gruen, Eyal Ofek, Anthony Steed, Ran Gal, Mike Sinclair, and Mar Gonzalez-Franco. Measuring system visual latency through cognitive latency on video see-through ar devices. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 791–799, 2020.

[12] Shree Krishna Sharma, Isaac Woungang, Alagan Anpalagan, and Symeon Chatzinotas. Toward tactile internet in beyond 5g era: Recent advances, current issues, and future directions. *IEEE Access*, 8:56948–56991, 2020.

[13] Forbes. Two killed in tesla crash with no driver at the wheel. `https://www.forbes.com/sites/jonathanponciano/2021/04/18/driverless-tesla-behind-crash-that-killed-two-in-texas-officials-believe/?sh=1aa3eff44824`, April 2021.

[14] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: Comparing public cloud providers. In *Proceedings of the 10th ACM*

SIGCOMM Conference on Internet Measurement, IMC '10, pages 1–14, New York, NY, USA, 2010. Association for Computing Machinery.

[15] Toke Høiland-Jørgensen, Bengt Ahlgren, Per Hurtig, and Anna Brunstrom. Measuring latency variation in the internet. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 473–480, 2016.

[16] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, (4):14–23, 2009.

[17] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[18] Amazon. AWS Global Infrastructure. `https://aws.amazon.com/about-aws/global-infrastructure/`.

[19] Microsoft. Azure Geographies. "`https://azure.microsoft.com/en-us/global-infrastructure/geographies/`", May 2021.

[20] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. Cloud provider connectivity in the flat internet. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 230–246, New York, NY, USA, 2020. Association for Computing Machinery.

[21] Kiryong Ha, Padmanabhan Pillai, Wolfgang Richter, Yoshihisa Abe, and Mahadev Satyanarayanan. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 153–166. ACM, 2013.

[22] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[23] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[24] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal de Lara, and Blesson Varghese. Defog: Fog computing benchmarks. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, SEC '19, page 47–58, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Inc. Amazon. Amazon lambda@edge. `https://aws.amazon.com/lambda/edge/`, 2021.

[26] Inc. Cloudflare. Cloudflare workers. `https://workers.cloudflare.com/`, 2021.

[27] Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. (how much) can edge computing change network latency? `https://mediatum.ub.tum.de/1609139`, 2021.

[28] Prathy Raman and Marcel Flores. Building out the basics with hoplets. In Oliver Hohlfeld, Andra Lutu, and Dave Levin, editors, *Passive and Active*

Measurement, pages 355–370, Cham, 2021. Springer International Publishing.
[29] Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. Pruning edge research with latency shears. https://mediatum.ub.tum.de/1574595, 2020.
[30] Maximilian Eder, Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. Surrounded by the clouds. https://mediatum.ub.tum.de/1593899, 2021.
[31] Vaibhav Bajpai, Anna Brunstrom, Anja Feldmann, Wolfgang Kellerer, Aiko Pras, Henning Schulzrinne, Georgios Smaragdakis, Matthias Wählisch, and Klaus Wehrle. The dagstuhl beginners guide to reproducibility for experimental networking research. *SIGCOMM Comput. Commun. Rev.*, 49(1):24–30, February 2019.
[32] B. Adelberg, H. Garcia-Molina, and B. Kao. Applying update streams in a soft real-time database system. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, SIGMOD '95, pages 245–256, New York, NY, USA, 1995. ACM.
[33] Junghoo Cho and Hector Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 117–128, New York, NY, USA, 2000. ACM.
[34] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis. Age and value of information: Non-linear age case. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 326–330, June 2017.
[35] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides. Effect of message transmission path diversity on status age. *IEEE Transactions on Information Theory*, 62(3):1360–1374, March 2016.
[36] M. Costa, M. Codreanu, and A. Ephremides. On the age of information in status update systems with packet management. *IEEE Transactions on Information Theory*, 62(4):1897–1910, April 2016.
[37] E. Najm and R. Nasser. Age of information: The gamma awakening. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2574–2578, July 2016.
[38] R. D. Yates and S. Kaul. Real-time status updating: Multiple sources. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 2666–2670, July 2012.
[39] R. D. Yates. Lazy is timely: Status updates by an energy harvesting source. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 3008–3012, June 2015.
[40] B. T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu. Age of information under energy replenishment constraints. In *2015 Information Theory and Applications Workshop (ITA)*, pages 25–31, Feb 2015.
[41] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff. Update or wait: How to keep your data fresh. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.
[42] Tan Bacinoglu and Elif Uysal-Biyikoglu. Scheduling status updates to

minimize age of information with an energy harvesting sensor. In *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1122–1126, 06 2017.

[43] X. Wu, J. Yang, and J. Wu. Optimal status update for age of information minimization with an energy harvesting source. *IEEE Transactions on Green Communications and Networking*, 2(1):193–204, March 2018.

[44] Songtao Feng and Jing Yang. Optimal status updating for an energy harvesting sensor with a noisy channel. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 348–353, 2018.

[45] B. Li, A. Eryilmaz, and R. Srikant. On the universality of age-based scheduling in wireless networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1302–1310, April 2015.

[46] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano. Minimizing the age of information in broadcast wireless networks. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 844–851, Sept 2016.

[47] Q. He, D. Yuan, and A. Ephremides. Optimizing freshness of information: On minimum age link scheduling in wireless systems. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–8, May 2016.

[48] Y. P. Hsu, E. Modiano, and L. Duan. Age of information: Design and analysis of optimal scheduling algorithms. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 561–565, June 2017.

[49] Ning Lu, Bo Ji, and Bin Li. Age-based scheduling: Improving data freshness for wireless real-time traffic. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc '18, pages 191–200, New York, NY, USA, 2018. ACM.

[50] Ahmed M. Bedewy, Yin Sun, Sastry Kompella, and Ness B. Shroff. Age-optimal sampling and transmission scheduling in multi-source systems. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc '19, pages 121–130, New York, NY, USA, 2019. ACM.

[51] B. Zhou and W. Saad. Optimal sampling and updating for minimizing age of information in the internet of things. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2018.

[52] V. Tripathi, R. Talak, and E. Modiano. Age optimal information gathering and dissemination on graphs. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2422–2430, April 2019.

[53] B. Yin, S. Zhang, Y. Cheng, L. X. Cai, Z. Jiang, S. Zhou, and Z. Niu. Only those requested count: Proactive scheduling policies for minimizing effective age-of-information. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 109–117, April 2019.

[54] K. M. Alam and A. El Saddik. C2ps: A digital twin architecture reference model for the cloud-based cyber-physical systems. *IEEE Access*, 5:2050–2062, 2017.

[55] Regional Internet Registries Statistics. World - autonomous system number

statistics - sorted by number. https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html. Accessed: 2021-08-20.
[56] man7.org. ping(8). "https://man7.org/linux/man-pages/man8/ping.8.html", 2021.
[57] man7.org. traceroute(8). "https://man7.org/linux/man-pages/man8/traceroute.8.html", 2021.
[58] NetBrain. Traceroute limitations explained. https://www.netbraintech.com/blog/limitations-of-traceroute/. Accessed: 2021-08-20.
[59] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, page 153–158, New York, NY, USA, 2006. Association for Computing Machinery.
[60] Yihua He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On routing asymmetry in the internet. In *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, volume 2, pages 6 pp.–, 2005.
[61] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. A measurement study of internet delay asymmetry. In Mark Claypool and Steve Uhlig, editors, *Passive and Active Network Measurement*, pages 182–191, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
[62] Ethan Katz-Bassett, Harsha V. Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter van Wesep, Thomas Anderson, and Arvind Krishnamurthy. Reverse traceroute. In *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 10)*, San Jose, CA, April 2010. USENIX Association.
[63] RN Staff. RIPE Atlas: A global internet measurement network. *Internet Protocol Journal*, 18(3), 2015.
[64] Vaibhav Bajpai, Steffie Jacob Eravuchira, and Jürgen Schönwälder. Lessons learned from using the ripe atlas platform for measurement research. *ACM SIGCOMM Computer Communication Review*, 45(3):35–42, 2015.
[65] RIPE NCC. Ripe atlas api reference. https://atlas.ripe.net/docs/api/v2/reference/, 2020.
[66] RIPE-NCC. Python client for ripe atlas api. https://github.com/RIPE-NCC/ripe-atlas-cousteau, 2020.
[67] RIPE-NCC. Official command line for ripe atlas. https://github.com/RIPE-NCC/ripe-atlas-tools, 2020.
[68] AWS. AWS Direct Connect Partners. "https://aws.amazon.com/directconnect/partners/".
[69] United Nations. World urbanisation prospects. "https://population.un.org/wup/DataQuery/", May 2021.
[70] Paddy Krishnan, Danny Raz, and Yuval Shavitt. The cache location problem. *IEEE/ACM transactions on networking*, 8(5):568–582, 2000.
[71] Lili Qiu, Venkata N Padmanabhan, and Geoffrey M Voelker. On the placement

of web server replicas. In *IEEE INFOCOM 2001*.
[72] Pavlin Radoslavov, Ramesh Govindan, and Deborah Estrin. Topology-informed internet replica placement. *Computer Communications*, 25(4):384–392, 2002.
[73] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. Pushing cdn-isp collaboration to the limit. *ACM SIGCOMM CCR*, 43(3):34–44, 2013.
[74] Lin Wang, Lei Jiao, Ting He, Jun Li, and Max Mühlhäuser. Service entity placement for social virtual reality applications in edge computing. In *IEEE INFOCOM 2018*.
[75] Jie Xu, Lixing Chen, and Pan Zhou. Joint service caching and task offloading for mobile edge computing in dense networks. In *IEEE INFOCOM 2018*.
[76] Bin Gao, Zhi Zhou, Fangming Liu, and Fei Xu. Winning at the starting line: Joint network selection and service placement for mobile edge computing. In *INFOCOM 2019*.
[77] CAIDA. CAIDA Archipalego (Ark) project. "`https://www.caida.org/projects/ark/`", 2020.
[78] Harsha V Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane: An information plane for distributed services. In *USENIX OSDI 2006*.
[79] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.
[80] Liang Wang, Suzan Bayhan, Jörg Ott, Jussi Kangasharju, Arjuna Sathiaseelan, and Jon Crowcroft. Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking. In *ACM ICN 2015*.
[81] ThousandEyes. Cloud Performance Benchmark. `https://www.thousandeyes.com/research/cloud-performance`, 2019.
[82] Todd Arnold, Ege Gürmeriçliler, Georgia Essig, Arpit Gupta, Matt Calder, Vasileios Giotsas, and Ethan Katz-Bassett. (how much) does a private wan improve cloud performance? In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 79–88. IEEE, 2020.
[83] Fabio Palumbo, Giuseppe Aceto, Alessio Botta, Domenico Ciuonzo, Valerio Persico, and Antonio Pescape. Characterizing cloud-to-user latency as perceived by aws and azure users spread over the globe. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019.
[84] Osama Haq, Mamoon Raja, and Fahad R. Dogar. Measuring and improving the reliability of wide-area cloud paths. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 253–262, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.
[85] Ondrej Tomanek, Pavol Mulinka, and Lukas Kencl. Multidimensional cloud latency monitoring and evaluation. *Comput. Netw.*, 107(P1):104–120, October 2016.
[86] Kapil Agrawal, Viral Mehta, Sundararajan Renganathan, Sreangsu Acharyya, Liting Zhao, Chakri Kotipalli, and Venkata N. Padmanabhan. Monitoring cloud service unreachability at scale. In *IEEE INFOCOM 2021*, 2021.
[87] Katerina Mania, Bernard D Adelstein, Stephen R Ellis, and Michael I Hill.

Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 39–47. ACM, 2004.

[88] Song-Woo Choi, Siyeong Lee, Min-Woo Seo, and Suk-Ju Kang. Time sequential motion-to-photon latency measurement system for virtual reality head-mounted displays. *Electronics*, 7(9):171, 2018.

[89] Kjetil Raaen, Ragnhild Eg, and Carsten Griwodz. Can gamers detect cloud delay? In *2014 13th Annual Workshop on Network and Systems Support for Games*, pages 1–3. IEEE, 2014.

[90] David L Woods, John M Wyma, E William Yund, Timothy J Herron, and Bruce Reed. Factors influencing the latency of simple reaction time. *Frontiers in human neuroscience*, 9:131, 2015.

[91] Google. Stadia. `https://stadia.dev/`, 2019.

[92] Statista. The statistics portal for market data. `https://www.statista.com/`, 2019. [Accessed: 2021-08-23].

[93] Srikanth Sundaresan, Nick Feamster, and Renata Teixeira. Home network or access link? locating last-mile downstream throughput bottlenecks. In *International Conference on Passive and Active Network Measurement*, pages 111–123. Springer, 2016.

[94] Haiqing Jiang, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Tackling bufferbloat in 3g/4g networks. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 2012.

[95] Lorenzo Corneo and Per Gunningberg. Scheduling at the edge for assisting cloud real-time systems. In *Proceedings of the 2018 Workshop on Theory and Practice for Integrated Cloud, Fog and Edge Computing Paradigms*, TOPIC '18, page 9–14, New York, NY, USA, 2018. Association for Computing Machinery.

[96] L. Corneo, C. Rohner, and P. Gunningberg. Age of information-aware scheduling for timely and scalable internet of things applications. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2476–2484, April 2019.

[97] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. Pruning edge research with latency shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, HotNets '20, page 182–189, New York, NY, USA, 2020. Association for Computing Machinery.

[98] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. A first look at commercial 5g performance on smartphones. In *Proceedings of The Web Conference 2020*, WWW '20, page 894–905, New York, NY, USA, 2020. Association for Computing Machinery.

[99] Lorenzo Corneo, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. Surrounded by the clouds: A comprehensive cloud reachability study. In *Proceedings of the Web Conference 2021*, WWW '21, page 295–304, New York, NY, USA, 2021. Association for Computing Machinery.

[100] Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. (how much) can

edge computing change network latency? In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2021.

[101] Speedchecker, Ltd. Speedcheker. `https://www.speedchecker.com/`, June 2021.

[102] Arvind Narayanan, Eman Ramadan, Rishabh Mehta, Xinyue Hu, Qingxu Liu, Rostand A. K. Fezeu, Udhaya Kumar Dayalan, Saurabh Verma, Peiqi Ji, Tao Li, Feng Qian, and Zhi-Li Zhang. Lumos5g: Mapping and predicting commercial mmwave 5g throughput. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 176–193, New York, NY, USA, 2020. Association for Computing Machinery.

[103] Measurement Lab. M-lab. `https://www.measurementlab.net/`, June 2021.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2074

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-452971

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2021

Paper I

Scheduling at the Edge for Assisting Cloud Real-Time Systems

Lorenzo Corneo
Uppsala University
Sweden
lorenzo.corneo@it.uu.se

Per Gunningberg
Uppsala University
Sweden
per.gunningberg@it.uu.se

ABSTRACT

We study edge server support for multiple periodic real-time applications located in different clouds. The edge communicates both with sensor devices over wireless sensor networks and with applications over Internet type networks. The edge caches sensor data and can respond to multiple applications with different timing requirements to the data. The purpose of caching is to reduce the number of multiple direct accesses to the sensor since sensor communication is very energy expensive. However, the data will then age in the cache and eventually become stale for some application. A push update method and the concept of age of information is used to schedule data updates to the applications. An aging model for periodic updates is derived. We propose that the scheduling should take into account periodic sensor updates, the differences in the periodic application updates, the aging in the cache and communication variance. By numerical analysis we study the number of deadline misses for two different scheduling policies with respect to different periods.

CCS CONCEPTS

• **Computer systems organization → Distributed architectures**; *Cloud computing*; *Sensor networks*;

KEYWORDS

Edge computing; cloud computing; sensor network; Industrial IoT; energy efficiency

ACM Reference Format:
Lorenzo Corneo and Per Gunningberg. 2018. Scheduling at the Edge for Assisting Cloud Real-Time Systems. In *TOPIC '18: Theory and Practice for Integrated Cloud, Fog and Edge Computing Paradigms 2018 Workshop, July 27, 2018, Egham, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3229774.3229777

1 INTRODUCTION

Cloud computing is increasingly becoming attractive for real-time systems. One attractive advantage is that major part of the control application execution can be off-loaded from the sensing and actuating devices to the cloud at the cost of communication. This off-loading is particularly important for battery operated devices.

TOPIC '18, July 27, 2018, Egham, United Kingdom

ACM ISBN 978-1-4503-5776-0/18/07…$15.00
https://doi.org/10.1145/3229774.3229777

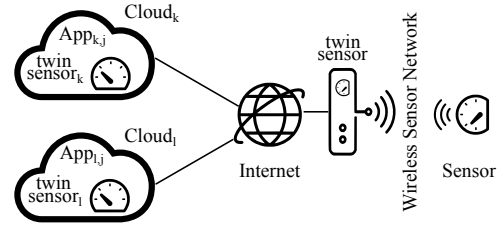


Figure 1: A real-time cloud based system with an edge server supporting an energy constrained sensor. The sensor data is replicated at the "twin sensors".

Although there are many scalability advantages with control logic in the cloud, the approach also has several consequences. First, since cloud servers are often provided at relatively long distances from the controlled system, there will be a significant communication delay between the actual sensing event and the application execution. In addition, a cloud server is designed to support many simultaneous users, which may cause application scheduling delays that may be significant. The data will age during the communication [6, 7], i.e. the information quality will decline. We here use the metric age of information to measure the quality, defined as the age of the data since it was generated. Eventually, the data will be too old and not valid anymore for a correct real-time response, i.e. the data has become stale and it will cause a real-time deadline miss.

Edge and fog computing [2] have been proposed to handle the distance problem of the cloud. The idea is to "delegate" time and safety critical functionality of the control logic to a server close enough to the controlled system, typically located at the network edge.

We consider real-time cloud-edge-sensor architectures composed of sensor devices located on a local wireless network with an Internet gateway, an edge server co-located with the gateway and a cloud infrastructure, hosting several applications that need timely data from the sensors. See figure 1. The edge acts as a smart caching proxy between the sensor devices and the clouds. If the edge caches sensor data, it can respond to multiple application requests at different clouds as long as the data is fresh enough for them.

In particular we consider periodic applications and battery operated sensor devices that are duty cycled in order to save energy. The caching can reduce the number of sensor accesses in order to save energy but at the cost of some aging in the edge that increases the probability of stale data. From energy conserving point of view it is desirable to have as long period as possible between duty cycles while still meeting the real-time requirements from the most stringent update periods of the applications.

The edge completely controls the communication to the sensor and caches the most recent sensor data value in a "digital twin sensor" (i.e. virtual sensor) [1], see figure, which is in turn exported to the corresponding twins at the clouds. The sensor value will age in the edge until it is renewed by a new sensor reading at a duty cycle. The cached value is pushed to the twins according to the periodicity of the applications. A problem with this approach is that the sensor cycle and the applications have different periodicity. This causes varying edge aging of data for the applications compared to when all applications do their own individual reading directly to the sensor. Aging increases the risk for stale data for correct real-time execution compared to direct access to the sensor and must be controlled at the edge. Our contributions are the following:

- Novel insights on how data ages in edge for periodic real-time cloud applications.
- An edge scheduling strategy that moves aging in the edge to aging in the cloud in order to decrease deadline misses.

We believe that we are among the first to study how a smart edge server could support cloud based real-time applications using duty cycled sensor devices.

In the following section we set a scenario and discuss related work. In section 3 our cloud-edge-sensor system is described and modelled with some simplifying assumptions. The model includes communication latencies and aging processes for multiple applications with different update frequencies. Section 4 describes when a scheduler in the edge may push data to the applications with respect to the aging at the cloud, edge and the periodicity of the applications. Then in the following section we evaluate two scheduling principles; a basic one determined by the periodicity of the applications and a smarter one that also take into the account the skew between the application periods. In section 5 we evaluate the two principles with numerical analysis with respect to how long time the data in the application is valid given the age of information and real-time constraints of the applications. Our purpose of the evaluation is to show the possible gain with a smart scheduling. The paper is finalized with a conclusion section.

2 SCENARIO AND RELATED WORK

As an illustrating scenario of a real-time cloud based system, consider autonomous forklift trucks in a factory, moving goods on the factory floor. To justify the need for delegating time critical functions to the edge, assume humans moving more or less randomly on the factory floor which requires the trucks to reliably and swiftly avoid hitting them. The anti-collision function is preferably located in the truck or a close-by server. On the other hand, a logistic program on where to pick up goods and to calculate the best route for the trucks is preferably done in the cloud since it needs all the truck positions and data about the location and destination of the goods. Still, the varying communication delays may cause stale data at the cloud.

Fog computing [2], or also called edge computing, is a paradigm that has been proposed to supply to the limitations of cloud computing. For example, fog computing can be used to enable highly responsive (e.g., low latency) cloud services, mask cloud outrages [9] and enforce privacy [3]. Edge computing is particularly used to enable mobility and one of the most successful implementations are cloudlets [10]. Another strong point of edge computing is that it can be used for offloading computation from constrained devices [4]. Previous effort in coupling data freshness and caching is presented in [11] in the context of Information Centric Networking.

The metric age of information is a concept that was firstly introduced by Kaul et al. [6] in the context of vehicular networks. The age of information is a metric for measuring data freshness and is formally defined as $\Delta(t) = t - u(t)$, where t is the current time and $u(t)$ is the time stamp of the freshest sensor update received by the cloud.

3 OUR CLOUD-EDGE-SENSOR SYSTEM

Consider a system consisting of a sensor device, an edge server, two or more clouds running different applications, as illustrated in figure 1. An application is periodically scheduled according to specifications, e.g a frequency of 4Hz (every 250ms). Hence, for periodic applications it is deterministically given when in real time the application should execute, given a start time for the first period.

Assume for simplicity, without sacrificing generality, only two applications App_k and App_l, one in each cloud. The applications are periodically executed with the frequencies f_k, f_l respectively, where $f_k > f_l$. Each cloud has a "digital twin sensor"[1], a data structure that is updated with the most recent value of the real sensor via the edge. An application will then conceptually read the twin in the same manner as if it would have been running on the sensor device or edge. This twin value is time-stamped by the sensor device when it is read, which means that an application can decide if it is stale or not according to its own clock and real-time specifications.[1] Intuitively the update frequency of the digital twin should be higher than the frequency of the application. But this is not enough, the age of the value has also to be within real-time boundaries. The value is aged by both the communication delays as well as waiting times for transmission resources.

How is the digital twin updated? The most straightforward way is the *push* model. The sensor device reads sensor data at wake-up periods and pushes the data to the edge twin for further pushes to the applications. Another model is *request/response*. Here the application instead requests an update to its twin when needed from the edge, i.e. a *pull* model. The edge in turn could also request an update from the sensor before delivering the response.

Communication. The communication path of data to the cloud has two distinct parts with different characteristics. The first part is between the sensor device and edge, typically it is a wireless sensor network optimized for small distances, energy conservation and predictable resource sharing. The second part of the path, between the edge and a cloud, is assumed to be a fully powered, high bandwidth network with regular Internet characteristics. The cloud service could be far away causing a substantial communication latency and when the network is shared there will also be considerable variance. The second part is normally dominating in terms of communication latency. The variance in the first part can in most circumstances be considered insignificant compared to the second.

[1]We assume that all clocks are enough synchronized and that clock drifts between synchronizations are not significant.

Table 1: Parameters for data aging and timing bounds.

$\Delta_a(t)$	Age of data at application period.
τ_Δ	Maximum acceptable age of data.
u_i	Time-stamp for the i:th sensor data.
r_i	Start of application for i:th data.
d_{se}	Packet delay between sensor and edge.
d_{ec}	Packet delay between edge and cloud.
$d_{v,i}$	Delay variance for i:th data.
t_{exec}	Tolerable length of application execution time.
σ	Delay before start of execution time.
t_c	Minimum required execution time.
v_i	Valid execution time for i:th data.

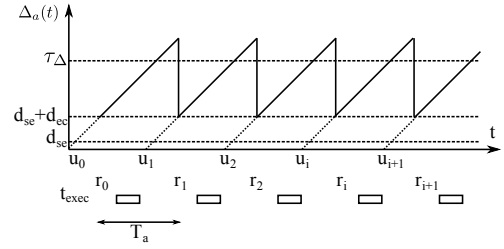
The communication latency can also be divided into a fixed, deterministic part and a varying part. The varying part includes data buffering, network contention and re-transmissions. Within the fixed we include propagation delays and transmission times assuming a known packet size. Since the wireless sensor network latency is both relatively small and typically time bounded we take the simplifying assumption to consider it as a fixed part. For this paper, packet losses are assumed to be handled by a reliable transport service. Packet re-transmissions may cause very long delays that breaks the age of information bounds. Other packet loss organization can be considered for real-time system, e.g. for periodic updates an idempotent approach may be favorable [5].

Edge. The edge server is fully powered and has the necessary computational capacity for processing and storing sensor data as well as servicing all requests from the applications. The edge also implements a digital twin that is updated by the sensor device and exported to the applications. It will initially request all applications' real-time requirements, including the update frequencies in order to optimize digital twin updates and sensor readings. Given these requirements, the edge can instruct the sensor device to change the length of the duty cycle in order to meet the aggregated requirements.

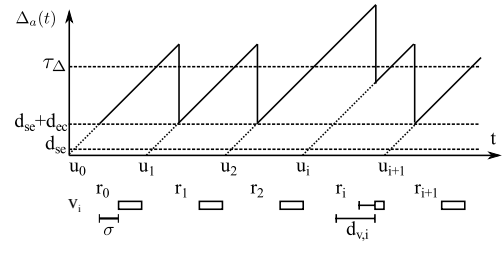
3.1 System behaviour

We now observe how sensor data ages in our system. Assume only one application in one cloud for the time being. The application is specified to run with frequency f_a, or with period T_a=1/f_a (from now on, we use period and frequency interchangeably). For a single application the sensor should produce data with at least the same frequency f_a. In other terms, the sensor will produce data periodically at time u_i, such that u_{i+1}=u_i+T_a. That data would then age on its way to the application. We assume u_0 to be the time when the sensor device starts producing updates.

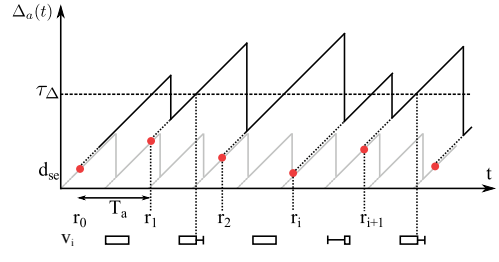
For further explaining the aging process, consider first the case with deterministic communication delays, as in figure 2(a). For a single application there is here no edge caching and the sensor data is directly propagated from the edge to the application. The delay part from the sensor to the edge is denoted d_{se} and the second part, from the edge to the twin in the cloud, is denoted d_{ec} in the figure. At time r_i=u_i+d_{se}+d_{ec} the data appears in the cloud twin with age $\Delta_a(t) = r_i$-u_i and the application is now enabled to execute. Intuitively, the application will perceive the age $\Delta_a(t)$ behaving like a "saw tooth" function that grows linearly with time and drops whenever a new update is received with a new time stamp.



(a) Aging of data with application period T_a



(b) Aging of data with variance delay



(c) Aging of data by edge server and variance delay

Figure 2: Aging process and validity period (below the graphs).

The execution could start immediately at r_i, or in the general case with a delay σ, i.e. at time $r_i + \sigma$. The maximum allowed execution time t_{exec} is given by the application requirements. This execution interval is illustrated in the figure as an unfilled bar. The parameter t_c specifies the actual computational time needed for the execution and must be less than t_{exec}. It allows the application scheduler to decide where in the t_{exec} interval the actual computation will take place, e.g. according to an Earliest Deadline First scheduling [8].

For periodic applications we therefore identify *two timing bounds that both must be met.* The first is that the execution must take place in the period t_{exec}, with start at $r_i + \sigma$. The second bound is on the age of data. Within t_{exec} the data must also be valid with respect to

age. In figure 2(a) we use τ_Δ to set the maximum allowed data age. It must be below τ_Δ at least t_c during a t_{exec} interval to be valid.

Both conditions must be fulfilled otherwise we have a deadline miss. Taken together they form a validity period, v_i, that the application only can execute in. For example, a system may specify that it should periodically produce results every 10 seconds but no later than 11 seconds but can tolerate up 2 second old data when it starts the execution.

In Figure 2(a), where we have no variance the design parameters τ_Δ, t_{exec}, σ and r_0=d_{se}+d_{ec} can hence be set to completely avoid deadline misses.

In Figure 2(b), delay variance in the second communication part is introduced as $d_{v,i}$. The other parameters are the same as for Figure 2(a). The data item for r_i is now exposed to an additional delay. As a result, the previous data in the twin will have to stay longer before being replaced, possibly beyond τ_Δ. In the figure, at time $r_i + \sigma$, the planned starting time of the execution, the existing value in the twin is too old and the application has to wait for an update that arrives at $r_i + d_{v,i}$. When $d_{v,i} > \sigma$ the valid execution period v_i hence becomes shorter. At some point it is shorter than t_c and then we have a deadline miss. Thus, with an increasing delay variance the probability of misses will also increase.

Algorithm 1 illustrates the pseudo-code for checking the validity of data. At the start of each execution period the application must first check if the age of the data in the twin sensor is within age boundaries during the time t_{exec}. If that is not the case the application is put on hold awaiting an update to the twin. When an update arrives the application once again reads the age of information and checks against the remaining execution time.

Algorithm 1 Pseudo-code of the application.

```
1:  function Application(τ_Δ, t_exec, t_c)
2:      if Δ_a(t) + t_c < τ_Δ then
3:          do_job()
4:      else
5:          δ = expected_time_next_update()
6:          if δ + t_c < t_exec ∧ Δ_a(t + δ) + t_c < τ_Δ then
7:              do_job()
8:          else
9:              misses++
10:     Application(τ_Δ, t_exec, t_c)
```

3.2 Multiple applications

We now consider the scenario in Figure 2(c) with two applications with periods T_k and T_l, where $T_k < T_l$. In this scenario the edge is active in caching sensor data. In order to meet the tightest update period the sensor duty cycle is consequentially set to T_k. This means that the twin at the edge will see an "aging saw tooth" with period T_k (see lower saw tooth in the figure). When the twin is updated with a new value the edge will immediately forward it to App_k, which will see its own "saw tooth", like the one for the single case.

The second application App_l has a longer period. In figure 2(c) we otherwise use the same parameters for App_l as for figure 2(b), including the delay variance to illustrate the impact of caching. In 2(c) the edge forwards data at every $r_i - d_{ec}$. The red dots are put at these times and they also indicates how much the data has aged before it is forwarded. The dashed lines from the dots are

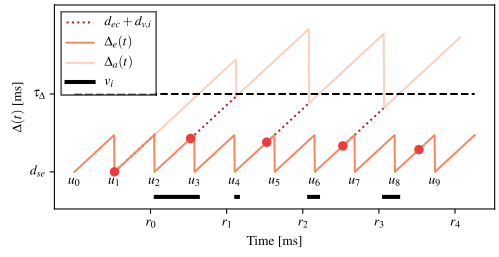


Figure 3: AITE - the edge pushes sensor updates to the digital twin according to the application period without considering aging at the edge and communication variance.

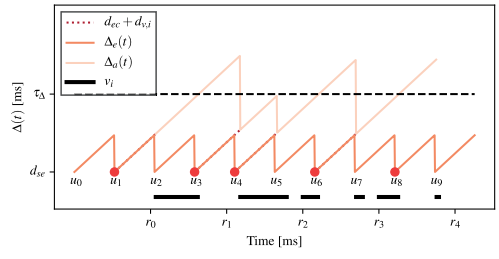


Figure 4: AITC - the edge pushes sensor updates to the application as soon as they get available.

communication times for the data to reach the cloud twin. It can be observed that the dots form a regular pattern, with equivalent distance on the edge aging teeth. This is a consequence of that T_l is not a perfect multiple of T_k. When T_l=nT_k. i.e. a multiple, all the red dots will have same age of information. The age of the data at the dots can be calculated as:

$$\Delta_l(r_i - d_{ec}) = d_{se} + [(i \cdot T_l) \bmod T_k] \quad (1)$$

The valid execution periods, v_i, are drawn at the bottom of figure 2(c). For event r_1, v_1 is shortened due an aging at the edge. For event r_i, v_i is shortened by an extended communication delay (i.e. caused by delay variance). The valid execution may be shortened in both ends and cause deadline misses.

4 SCHEDULING POLICIES

We now describe two scheduling policies in which the edge will provide mechanisms for sending sensor updates to the digital twins in the cloud according to applications execution periods.

The first scheduling policy is called "Age in the Edge" (AITE) and is designed to deliver sensor updates to an application within the execution time t_{exec}. Figure 3 shows how the age of sensor updates is evolving for this policy. In the lower part of the graph, we recognize the saw teeth at the edge and the validity periods.

The main drawback with AITE is the aging at the edge. The data is already aged when pushed to the cloud and it may then also

be exposed to network congestion prolonging the aging further. An intuitive solution to improve the situation would be to more frequently push updates towards the digital twin. For example, all sensor updates could be forwarded to the twin. This solution will however increase the number of updates in the network which is not desirable.

The second scheduling policy is named "Age in the Cloud" (AITC) and is illustrated in figure 4. It is designed to send the updates at the beginning of a saw tooth rather than let them age at the edge before pushing them as in AITE. Since it is predictable which updates will be aged at the edge it is possible to reschedule the updates so that there will always be a fresh value. Such a predicted aged update could be sent as soon as it arrives, which is the case for u_4 or the update could be delayed to the next fresh value as for u_3. When it is sent earlier the update will age in the cloud instead but has the advantage that it may tolerate communication variance better. When it is sent later, the data is fresher and it may tolerate better the age boundary even if it arrives later in the t_{exec} period.

In AITC the validity period will increase compared to the AITE scenario. We show differences between AITE and AITC in figures 3 and 4 respectively.

5 EVALUATION

In this section we evaluate the aforementioned AITE and AITC scheduling policies from the point of view of three different metrics. First of all, we observe the experienced age at the cloud $\Delta_c(t)$ upon reception and at application execution $\Delta_a(t)$. Then, we analyze the amount of time an update is usable before breaking the deadlines τ_Δ, $\tau_\Delta - \Delta_c(t)^{+2}$. Finally, we discuss the validity execution interval of an update, v_i. These metrics are important as they reflect application's execution flexibility and probability of experiencing real-time misses.

In our evaluation setup, the sensor device produces a new value once every 100ms. We assume a fixed communication delay between the sensor device and the edge of $d_{se} = 1$ms. The cloud infrastructure is placed at a transmission and propagation distance of $d_{ec} = 100$ms from the edge. The communication is affected by an additional delay with variance ($d_{v,i}$), according to a Pareto distribution with an expectation value of 14ms. We arbitrarily selected Pareto since it is commonly used for Internet delay distributions, but the same general behavior and main conclusions would apply to other distributions. The application in the cloud is executed once every $T_a = 190$ms with an execution time of $t_{exec} = 152$ms starting from the beginning of the application period (i.e. $\sigma = 0$) and $t_c = 0$. The upper bound on age of information for this application is set to $\tau_\Delta = 210$ms.

We perform numerical evaluations based on the aging model, the above parameters and the two scheduling principles described in previous section, illustrated in figures 3 and 4. We collect the results from 5000 application readings of the twin sensor in the cloud. These 5000 readings are affected by 5000 delay samples from the Pareto distribution. The maximum value of these samples is 161ms.

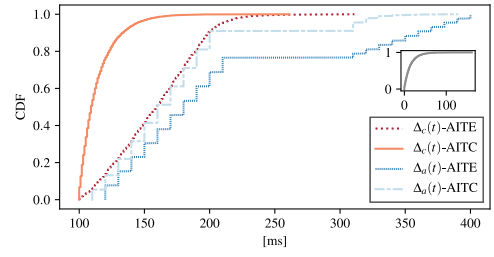


Figure 5: CDF of age of information for AITE and AITC upon updates reception at the cloud, $\Delta_c(t)$, and at application, $\Delta_a(t)$. On the right the variance distribution is shown.

5.1 Numerical results

Figure 5 shows the cumulative distribution function of the age of information of the updates when they are received at the digital twin in the cloud, $\Delta_c(t)$, and when the application is executed, $\Delta_a(t)$. First of all, we discuss $\Delta_c(t)$. The distribution of the age of information for AITE is basically linear due to the misalignment between the sensor and application periods. This misalignment, in the long run, forces the age of information at the application to be uniformly distributed throughout the aging saw tooth. Furthermore, communication variance (showed in the small box at the right) is additive and extends age of information's upper-bound. The distribution of AITC, on the other hand, follows closely the variance distribution since all the updates are sent immediately upon reception at the edge. For this reason, the age of information $\Delta_c(t)$ follows the variance $d_{v,i} + d_{se} + d_{ec}$. These two CDFs confirm what can be expected about the relative parameters of edge aging and expected variance.

From the point of view of the application, $\Delta_a(t)$, AITE shows the worst performance. In fact, the misalignment of different periods impacts even more the already degraded distribution of $\Delta_c(t)$ for AITE. On the other hand and despite the same misalignment problem, AITC performs better than AITE. Intuitively, the better freshness exhibited by "$\Delta_c(t)$-AITC" approximately translates the same "$\Delta_a(t)$-AITC" closer to the origin on the X-axis. The previous insights are crucial to properly dimension real-time requirements, including τ_Δ.

Figure 6 presents the cumulative distribution function of the amount of time an update is usable before breaking the timing requirement τ_Δ, "$\tau_\Delta - \Delta_c(t)^+$". In other words, we can see for how long one update will be valid before it gets too old for correct processing. To better understand this plot, we provide the insight that the more the curve is distributed on the right, the better. In fact, the higher the numbers on the X-axis, the longer the validity time on age of information. We observe that, for AITE, roughly 4% of the updates is already too old for usage at the arrival at the cloud (value 0 in the plot). Maybe without surprise, AITC obtains better results. The validity in AITC is better because the updates are forwarded upon reception. Nonetheless, having a larger margin to τ_Δ is not

[2]Only positive values allowed, whenever negative the returned value is zero.

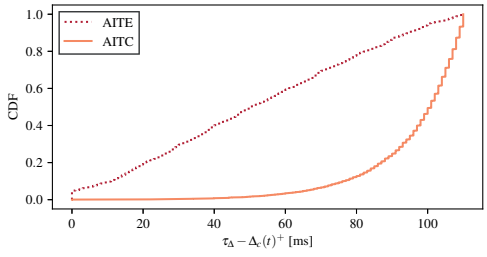


Figure 6: CDF of the amount of time an update is usable before breaking timing requirement τ_Δ.

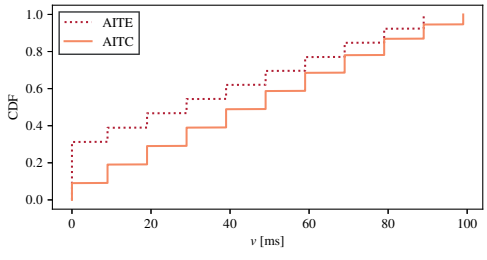


Figure 7: Validity v of the updates at the application.

enough since the update must also be valid during the application's execution time t_{exec}. This leads us to our last proposed evaluation metric.

Figure 7 shows the cumulative distribution function of v, the validity interval of an update within the application's execution time t_{exec}. As for the previous plot, the more the CDF is distributed on the right, the better. Also in this case, AITE performs worse than AITC. For instance, 34% of the updates are not valid with respect to t_{exec} (CDF value 0). This confirms the hypothesis made in section 3.1 where we stated that both aging at the edge before transmission and communication variance reduce the validity v for such application. In contrast, AITC outperforms AITE as it is designed to deliver the freshest updates as close as possible to the application execution. From the figure we observe that we obtain only 9% of misses and we also have significant improvement in the validity of the received updates at the application.

5.2 Discussion

We evaluated two different scheduling policies, namely, AITE and AITC. We demonstrated that aging in the cloud is better than aging in the edge, especially when updates must travel through the Internet and are exposed to non-negligible communication variance. Furthermore, scheduling with knowledge about misalignment between periods helps real-time applications in meeting their timing requirements without the need to emit updates with higher frequencies, which would increase the network traffic and reduce sensors battery life.

One may argue that the communication model used is too simplistic, but this is done on purpose. In fact, the system already relies on numerous parameters spread over several distributed components. Adding further complexity will mask the base overall functioning of the system impacting the understanding of the proposed challenges and solutions.

6 CONCLUSION

We study the behaviour of a distributed real-time systems composed of sensor devices, wireless sensor networks, edge server(s) and a remote cloud infrastructure. We make use of the age of information as a main metric for evaluating the freshness of sensor updates at cloud based periodic applications with timing requirements. For periodic applications we identify two validity timing bounds on sensor data that both must be met. The first is that the data must not be too old for the application. The second is that even if the data is fresh enough it must also be available within the specified execution time interval of the application to be valid. We showed that an edge scheduler can play a fundamental role for delivering fresh data within a specified execution interval of a periodic application. We evaluated numerically two scheduling policies providing insight on how to improve the validity time interval of updates and we discussed trade offs among the several parameters of the system. The purpose of the evaluation is to show the potential gain of scheduling updates to applications with mismatched periodicity and duty cycled sensors. Even if the evaluation is not comprehensive we draw the conclusion that there are considerable possible improvements a designer of an edge-cloud based real-time system should be aware of. A more comprehensive study scaling for several applications, cloud services, more realistic communication delays and different scheduling parameters remains for future work.

7 ACKNOWLEDGEMENT

This work was supported by the Swedish Foundation for Strategic Research under the project "Future Factories in the Cloud (FiC)" with grant number GMT14-0032.

REFERENCES

[1] K. M. Alam et al. 2017. C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems. *IEEE Access* (2017).
[2] Flavio Bonomi et al. 2012. Fog Computing and Its Role in the Internet of Things *(ACM MCC '12).*
[3] Nigel Davies et al. 2016. Privacy Mediators: Helping IoT Cross the Chasm *(ACM HotMobile '16).*
[4] Kiryong Ha et al. 2014. Towards Wearable Cognitive Assistance *(ACM MobiSys '14).*
[5] Pat Helland. 2012. Idempotence is Not a Medical Condition. *Commun. ACM* (2012).
[6] S. Kaul et al. 2011. Minimizing age of information in vehicular networks *(IEEE SECON 2011).*
[7] S. Kaul et al. 2012. Real-time status: How often should one update?. In *2012 Proceedings IEEE INFOCOM.*
[8] C. L. Liu et al. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* (1973).
[9] M. Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* (2017).
[10] M. Satyanarayanan et al. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* (Oct 2009).
[11] S. Vural et al. 2017. Caching Transient Data in Internet Content Routers. *IEEE/ACM Transactions on Networking* (April 2017).

Paper II

Age of Information-Aware Scheduling for Timely and Scalable Internet of Things Applications

Lorenzo Corneo, Christian Rohner and Per Gunningberg
Department of Information Technology, Uppsala University, Sweden
Email: lorenzo.corneo@it.uu.se, christian.rohner@it.uu.se, per.gunningberg@it.uu.se

***Abstract*—We consider large scale Internet of Things applications requesting data from physical devices. We study the problem of timely dissemination of sensor data towards applications with freshness requirements by means of a cache. We aim to minimize direct access to the possibly battery powered physical devices, yet improving Age of Information as a data freshness metric.**

We propose an Age of Information-aware scheduling policy for the physical device to push sensor updates to the caches located in cloud data centers. Such policy groups application requests based on freshness thresholds, thereby reduces the number of requests and threshold misses, and accounts for delay variation. The policy is incrementally introduced as we study its behavior over ideal and more realistic communication links with delay variation. We numerically evaluate the proposed policy against a simple yet widely used periodic schedule. We show that our informed schedule outperforms the periodic schedule even under high delay variations.

I. Introduction

Internet of Things (IoT) enabled applications in the domains of health care, environmental and building monitoring must rely on timely and fresh information gathered from a wide plethora of sensing devices. Age of Information (AoI) [1], [2], the age of the data since it was generated, is a metric indicator of data freshness. AoI is formally defined as $\Delta(t) = t - U(t)$, where t is the current time and $U(t)$ the time stamp of the most recent information update.

Since its introduction in [2], the AoI concept has been studied with different approaches and in different contexts. AoI minimization problems have been broadly studied in the field of queuing theory in [2]–[8] and in the context of energy harvesting systems, where energy is intermittent due to uncertain environmental conditions [9]–[14]. In wireless communication links, AoI-aware scheduling policies have been proposed in [15]–[19]. Furthermore, AoI has been used for applications in the domain of cloud gaming [20] and the dissemination of content updates in mobile social networks [21].

Cloud computing [22] goes hand in hand with IoT. Cloud servers provide seemingly unlimited storage and computational power, is geographically distributed and accessible all around the globe. Cloud computing is particularly attractive for off-loading storage and computation from battery and battery-free operated sensing devices. Although there are many scalability advantages, there is a significant communication latency between the actual sensing event and the IoT applications in the cloud. To overcome this latency limitation, edge and fog computing [23] have been proposed to move cloud functionality closer to the physical devices, typically at the edge of the network [24]. Time and safety critical functionality can then be performed with lower latency.

Nonetheless, the intrinsic nature of IoT applications faces scalability challenges itself. In fact, the sensing device is a bottleneck as it is the main entity that generates information for the IoT applications. In addition, sensing devices may be energy constrained and should not have to reply to each and every application request. Cloud and edge infrastructures linking applications with the sensing devices are therefore exposed to a trade-off to either provide fresh information from the devices, by extensively accessing them, or provide applications with cached information that are aged.

In this paper, we present an AoI-aware scheduling policy aimed to solve the problem of 1-to-many information dissemination from sensors to several remote applications. In particular, we focus on reducing sensor energy consumption yet satisfying applications timing requirements in the presence of varying communication delays. We consider a system architecture composed by cloud servers, edge servers and physical systems of sensing devices. We use the storage and computational power provided by the cloud as a mean for content distribution of sensor updates towards several remote applications. Then, the edge aggregates applications subscriptions to sensor updates coming from geo-distributed cloud instances. As a result, the edge server generates "smart" schedules to be downloaded to the sensor devices for duty-cycling purposes. Our contributions for this work are:

- A novel AoI-aware scheduling policy for sensor networks which reduces energy consumption while still ensuring fresh enough information towards several remote applications.
- A method that allows remote applications to meet their freshness requirements despite the effect of delay variations, especially between edge server and cloud.

The remainder of this paper is organized as follows. Sec. II and III describe the system and performance metrics. Sec. IV and V incrementally introduce scheduling policies in ideal and delay constrained conditions, respectively. The performance of the AoI-aware schedule is evaluated and discussed in Sec. VI.

II. System

We consider the system illustrated in Fig. 1 that involves sensing devices, an edge server and one or more cloud data centers hosting different applications.

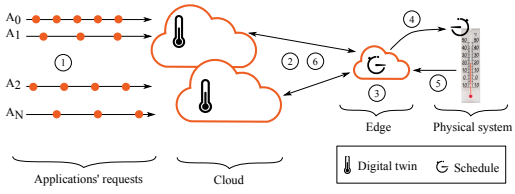


Fig. 1. System architecture.

① Generic **applications** with requirements on data freshness subscribe to the closest (in terms of latency) cloud instance. They specify the interest in particular sensors, an execution schedule and an upper-bound on freshness of the information. We consider applications that run periodically but can tolerate some variance in the scheduled execution times. Examples of such applications are monitoring and control systems.

② The **cloud** instances export sensor readings through an artifact commonly called "**digital twin**" sensor [25]. The digital twin allows the applications to access cached copies of sensor readings as if they were accessed directly at physical sensing devices, with some aging. The use of such artifact is motivated by latency. In fact, as the communication latency between the applications and the physical sensors may be significant, a *pull* update strategy may not be the best option for applications with strict timing requirements. As a result, we propose to update digital twins with a *push* policy from the physical sensing devices and edge device. Digital twins can be replicated at multiple cloud instances in order to minimize delay for data dissemination towards the applications. Furthermore, the subscriptions and the requirements on digital twin sensors are forwarded to an edge device on behalf of the applications.

③ The **edge** device is located close to the sensor network. As first responsibility, it receives from cloud instances all the subscriptions and requirements of the applications. Then, it aggregates all the requirements and combines them together creating a "smart" schedule. The aim of this schedule is to instruct the sensors on *when* to push updates toward the edge server. Such schedule is able to minimize the number of sensor update transmissions yet providing the required level of information freshness. Furthermore, it accounts for communication delay towards the cloud infrastructures. ④ After being computed, the schedule is pushed towards the physical sensor.

The **physical sensing devices** consist of a sensor network that provides sensor readings. The sensors in the network are duty-cycled in order to save energy and prolong battery lifetime. ⑤ The duty-cycling is dictated by the schedule received by the paired edge device. ⑥ Once the edge server receives the aforementioned sensor updates, it will push them towards the respective digital twins in the cloud. Without loss of generality, we consider sensor updates from one particular sensor to a possibly large number of applications.

III. Model

A. Notation and Parameters

Let $A = (a_1, a_2, ..., a_N)$ be the set of the N applications with timing requirements in the system. These applications are periodically executed and we indicate these periods with the set $T = (T_1, T_2, ..., T_N)$, where the subscript refers to applications in set A.

The applications in set A request sensors information at a cloud that provides digital twin services. The set of $M \in \mathbb{N}_+$ cloud instances is defined as $C = (c_1, c_1, ..., c_M)$. We indicate the latency (one way delay) between application a and sensing devices as $d_{a,s}$. To be noticed, every application specifies upper-bounds on AoI, τ, and may accept a delayed response by ϵ time units and wait for a fresher update. We make a simplifying assumption that all the applications have the same freshness threshold τ.

Let $x_a = \{t_0 + i \cdot T_a : i \in \mathbb{N}_+\}$ be the set of periodic execution times of application $a \in A$ with period $T_a \in T$ and start time t_0. As the application subscribes its interest in sensors information to a cloud $c \in C$, the latter knows the execution schedule of such application (x_a). Additionally, we define the set of applications subscribed to a particular cloud $c \in C$ as A_c, where $A_c \subseteq A$.

Finally, we define the execution schedule of all the applications subscribed at a cloud $c \in C$ as $X_c = \bigcup_{\forall a \in A_c} x_a$. To notice, all the elements in X_c are sorted such that $x_i < x_{i+1}, \forall x \in X_c \wedge i \in \mathbb{N}_+$. For simplicity of notation and readability, we assume that all the applications subscribe to the same cloud and we therefore avoid suffix c throughout the paper (i.e. $A_c \to A, X_c \to X$).

B. Evaluation Metrics

We take into account the following four evaluation metrics:

- The ratio between the number of transmitted sensor updates and the total number of application requests in the system, ρ, is defined as:

$$\rho = \frac{\text{\# sensor updates}}{\text{\# applications requests}} \tag{1}$$

- The average Age of Information of all the applications in the system, Δ_A, is formally defined as:

$$\Delta_A = \frac{1}{|X|} \sum_{i=0}^{|X|} \Delta(x_i) \tag{2}$$

  where x_i is the i:th scheduled application execution and $\Delta(\cdot)$ is the instantaneous age of information defined as $\Delta(t) = t - U(t)$, where t is the current time and $U(t)$ the update's generation time [2].
- The average response time of all the running applications, ReT, is defined as

$$\text{ReT} = \frac{1}{|X|} \sum_{i=0}^{|X|} \hat{x}_i - x_i \tag{3}$$

where $\hat{x_i}$ is the actual execution time of x_i and we also assume that $\hat{x_i} > x_i$. The main idea behind this metric is that the applications may have *delay budgets* and are able to tolerate longer time for the response to a request with the aim to obtain better AoI. The behavior of response time is further explained in the next section.

- The number of deadline misses is delivered by a threshold function that measures the number of stale updates that have been read by the applications. We formally define this metric as the cardinality of the elements older than τ, for all the applications requests:

$$\text{misses} = \frac{|\{x : x \in X \wedge \Delta(x) > \tau\}|}{|X|} \tag{4}$$

It is straightforward to extend the formula to a multi-threshold case by adding individual application identifier suffixes to thresholds (e.g., τ_a with $a \in A$). The resulting number of misses would be the summation of individual applications misses.

IV. Scheduling Policies

In this section we incrementally introduce scheduling policies to improve AoI. Schedules are created at the edge server after receiving timing requirements of the applications, forwarded by the cloud. Eventually, individual schedules are downloaded into sensing devices. To start with, we assume ideal communication links without propagation delay and packet loss. Therefore, the delay between applications and sensors $d_{a,s} = 0\,\text{ms}$ (including between edge and cloud) and, hence, AoI can reach $0\,\text{ms}$ as well.

A. Age of Information Optimal, π^*_Δ

π^*_Δ is a scheduling policy that delivers optimal AoI. It achieves this by matching every application execution in X with a sensor update. The Age of Information Optimal scheduling policy is formally defined as:

$$\pi^*_\Delta(X) = \bigcup_{x \in X} \{x - d_{a,s}\} \tag{5}$$

To put it in simple words, we can interpret equation 5 as a scheduling policy that instructs sensors to send one update for every application execution. The update should be sent in advance of a quantity equal to the delay between the sensing device and the application. Assuming a link with no delays, it is possible to skip the addend $d_{a,s}$. As a result, this will always deliver $\Delta_A = 0\,\text{ms}$. As a consequence, the ratio between applications requests and sensor reading transmissions is $\rho = 1$ and $\text{ReT} = 0\,\text{ms}$.

B. Periodic Updates, π_P

While the Optimal AoI policy requires a sensor update for each request, it is possible to schedule fewer updates if one update can satisfy multiple requests, given that they are all within the acceptable AoI. That is, we trade an increased average AoI against fewer updates.

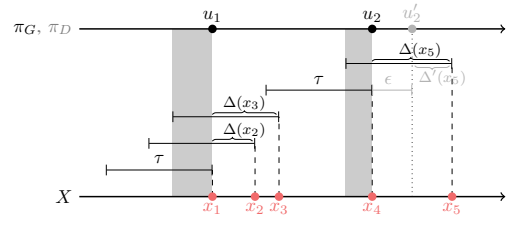


Fig. 2. Schedule π_G: Reducing sensor updates by exploiting intersections between τ segments. The scheduled sensor updates are appropriately scheduled at the end of the intersections (illustrated as gray areas). In light gray, an example of Schedule π_D for $x_{4,5}$ with a delayed update u'_2 to reduce the age of information for x_5.

Periodic delivery of sensor updates is one of the most common approaches in practice. We formally define a periodic scheduling policy as:

$$\pi_P(T_s) = \{t_0 + i \cdot T_s : i \in \mathbb{N}_+\} \tag{6}$$

where T_s is the updating period of sensor S and t_0 is the time of the first update.

What period should be chosen? The objective is to decrease ρ while keeping $\Delta(\cdot)$ bounded in the interval $[0, \tau]$. Hence, there is an upper-bound on the period when a request will provide the applications with stale data, i.e., older than τ.

The scheduling policies presented so far, namely, π^*_Δ and π_P, will be used as base lines for comparison between two other forthcoming policies that trade more effectively ρ against AoI.

C. Grouping Window, π_G

In the periodic policy, we adjust the update period to accommodate as many applications as possible for each update. In the Grouping Window policy, we will instead maximize the number of applications accessing the same update by breaking the periodicity between the updates.

How do we assign the times for the updates within a group of requests? Fig. 2 illustrates how they are grouped by using τ. We assume that all the applications in A share the same maximum acceptable τ.

In the figure, all the scheduled application executions in X are represented as (red) dots on the time line at the bottom. At the top time line, the corresponding sensor updates are marked (in black). Every execution can tolerate an AoI between 0 and τ. All the (red) dots within the shaded grey areas (grouping windows) can share the same sensor update. A schedule π_G is then composed by $\{u_1, u_2\}$ and so on. Intuitively, the algorithm finds a window that includeds as many requests as possible. These applications can be accommodated by a single update and the delivery of stale data is then avoided. In the figure, the updates are reduced from five to two. This algorithm is expected to do a better balancing between ρ and AoI than π_P.

Algorithm 1: Grouping Window with Delay Budget

Data: X, the schedule of applications executions
τ, applications timing requirement
ϵ, applications delay budget

1 $S \longleftarrow \emptyset$;
2 $i \longleftarrow 0$;
3 **while** $i < |X|$ **do**
4 $j \longleftarrow 1$;
5 **while** $X_{i+j} \leq X_i + \tau \wedge i + j < |X|$ **do**
6 $j \longleftarrow j + 1$;
7 **if** $\epsilon > 0 \wedge \exists X_a \in [X_i, X_i + \epsilon]$ **then**
8 $S \longleftarrow S \cup \{X_{i+|X_a|}\}$;
9 **else**
10 $S \longleftarrow S \cup \{X_i\}$;
11 $i \longleftarrow i + j$;
12 **return** S

D. Grouping Window with Delay Budget, π_D

In this scheduling policy we will extend the Grouping Window policy, π_G, by using the observation that many periodic application executions can tolerate a little delay, when getting the data from the digital twin. We denominate this policy *Grouping Window with Delay Budget*, π_D. The small delay acts as a delay budget for getting the data. The digital twin can deliberately postpone the response to the application up to the delay budget, we here call it ϵ. Why should we use such delay budget? One answer is that it allows a scheduler to detect that a digital twin can deliver a new update arriving within the ϵ interval. Thus, the application will experience better AoI than in the case of delivering the previous update. As a result, we can trade some longer response time, for each application and up to ϵ, against lower average AoI.

The operational difference between π_G and π_D is shown in Fig. 2. The application running at time x_4 can tolerate up to ϵ delay and we exploit this for delivering better AoI at time x_5. Algorithm 1 shows a pseudo code implementation of the Grouping Window with Delay Budget. At line 5-6, we define the grouping window including in it all the applications executions in the interval $[X_i; X_i + \tau]$. Then, at line 7, we check whether there are other applications scheduled starting from the first application up to ϵ. If that is the case, line 8, we delay the sensor update up to the latest execution in the aforementioned interval ($X_{i+|X_a|}$ in the code). If there are no additional application executions in $[X_i, X_i + \epsilon]$, the sensor update is scheduled at X_i. In the remainder of the algorithm, we create the schedule and we manipulate the indexes of set X so to process it from beginning to end.

E. Relation between schedules

The presented schedules are related to each others. π_G with $\tau = 0$ corresponds to π^*_Δ since no request can be grouped. Likewise, π_D with $\epsilon = 0$ corresponds to π_G and, for this reason, Algorithm 1 is valid also for π_G. For instance, π_G can be obtained by removing line 7-9 from Algorithm 1.

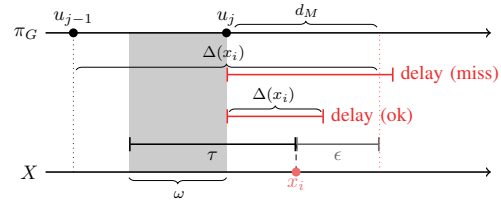


Fig. 3. The choice of the grouping window in π_G depends on the delay variation. We chose the grouping to minimize misses (i.e., $\Delta_A > \tau$), that is, the margins from maximum acceptable age of information plus delayed reply has to be larger than the grouping window plus a safety margin to account for the delay distribution: $\tau + \epsilon > \omega + d_M$.

V. Communication Delays

We assumed so far ideal communication channels without propagation delay and errors. In this section we discuss the impact of propagation delays on the performance of the schedules and introduce a delay-aware adaptation of the previously introduced policies. Delays can partly be compensated by adapting the schedule if their variation is small and can be tracked by an appropriate algorithm. In this paper, however, we focus on the worst case scenario when the delay variation is highly random. As a result, such variations cannot be tracked, yet we can compensate for the minimum delay.

We distinguish between two communication segments: between edge and cloud, and cloud and applications as they affect AoI in a different context. The delay between edge and cloud affects a set of grouped application requests. On the other hand, the delay between cloud and applications affects individual application requests. Studying these delays separately is motivated by the fact that we assume compensation for constant delay components; a dominating delay variation among cloud and applications can be considered a general case covering delay variation on both segments, and a dominating delay variation between edge and cloud includes the scenario where applications are hosted in the same cloud as the digital twin.

A. Digital Twin Strategies

The digital twins in the cloud are aware of the schedules produced by the edge and can measure delays in order to obtain delay averages and variance. On the basis of this knowledge, they can estimate when sensor updates are likely to come. In the best case, a twin could calculate a percentile, say 95:th, of the delay distribution and use that together with ϵ and τ to see if it is worth waiting for an update.

Given this knowledge, the twin can decide to either immediately deliver the last update to the application or decide to wait, if it is likely that a better update will come within the ϵ time. The choice depends on which of the previous and the

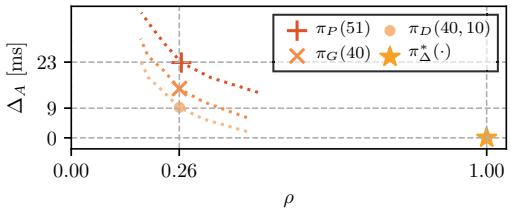


Fig. 4. Average age of information Δ_A against ratio samples/requests (assuming perfect links).

forthcoming updates that are expected will have the lowest AoI.

Furthermore, for unpredictable communication delays there is always a risk for stale data and out-of-order updates. We assume digital twins to be preemptive, meaning that outdated updates are automatically discarded.

For benchmarking our proposed policies we also constructed an "Oracle digital twin" who knows the exact communication delay for each individual update, now as well as in the future. This means that the Oracle can predict when all updates will arrive to the twin and always decide whether it is better to wait for an update or to deliver the previous one in order to minimize the AoI.

B. Delay between Edge and Cloud

When communication delays affect the network path between an edge and a cloud, all the applications subscribed to a particular digital twin will be out of synchronization with respect to the updates. For example, assuming a temporary delay at the path, the sensor updates will be received by all the applications later than expected and they may all experience stale information. Nonetheless, in case of frequent sensor updates, this misalignment may be insignificant since the application will get fresh enough data anyway in the near future. We propose a simple yet effective way to handle communication delays between edge and cloud in Sec. V-D.

C. Delay between Cloud and Applications

The delay between the twin in the cloud and the application will not be discussed further besides the observation that it only affects individual applications. The digital twin is aware of these delays (e.g., via round-trip time estimation using stamps). When the delays become significant the digital twin can notify this to the edge device that will then re-arrange the schedule. The method proposed in the next section can be applied in that case.

D. Miss Avoidance Grouping Window, π_ω

The Grouping Window policy suffers from communication induced delays in two ways. First, an update which is delayed for more than ϵ will lead to a deadline miss for that application. A deadline miss will also happens when an update arrives after the end of the grouping window. With this policy, even a small delay variations may significantly impact the number of misses. We now propose a method for mitigating this undesired sensitivity and suggest boundaries on the maximum number of deadline misses. The main idea is to reduce the length of the grouping window with a "safety margin" that accounts for the variations in the delays. Intuitively, the probability of misses will decrease with a smaller window.

Knowing the extent of delay variations and/or distributions would assist system's designers to target (and obtain) a limited amount of deadline misses while trading off with higher ρ. For example, assume that we aim at 5% misses over the total number of application requests. In order to achieve this, we must calculate the 95:th percentile of the delay distribution. The aforementioned safety time is then set at the 95:th percentile. For convenience in this paper we will call the safety margin d_M.

We now re-define the size of the grouping window with the safety margin d_M for varying delays as follow:

$$\omega = \tau + \epsilon - d_M \tag{7}$$

A graphical representation of the scheduling policies is shown in Fig. 3. It should be observed that the physical device should now send the update in advance of a quantity equal to $d_M - \epsilon$. Assuming that a schedule S' is the output of Algorithm 1 (with $\tau = \omega$ and $\epsilon = 0$[1]), a new schedule, S, with safety margin d_M can then be formalized in the following way:

$$S = \{s + \epsilon - d_M : s \in S'\} \tag{8}$$

Intuitively, while decreasing the number of misses, the new schedule will provide higher average AoI.

VI. Evaluation and Discussion

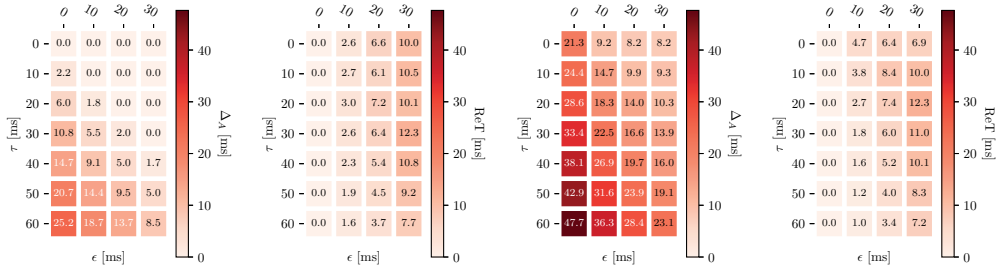
In this section we evaluate numerically the metrics introduced in Sec. III-B comparing scheduling policies presented in Sec. IV and V. For every simulation instance we provide $5\,\text{min}$ of simulated time that provides enough statistical relevance as we operate at milliseconds granularity. If not explicitly mentioned, we assume N = 10 application with periodicity randomly selected in the interval $[100, 200)$ ms. When we account for delays, we assume samples from a Pareto distribution [26] with shape $a = 1.672$ and scale $m = 5$.

Whenever the periodic policy, π_P, is compared to one of our proposed schedule, we choose the updating period, T_s, to correspond to the mean inter-arrival time between the updates of such schedule. As a result, the periodic schedule and the proposed one will have approximately the same ρ.

A. Age of Information vs. Ratio Updates-Requests

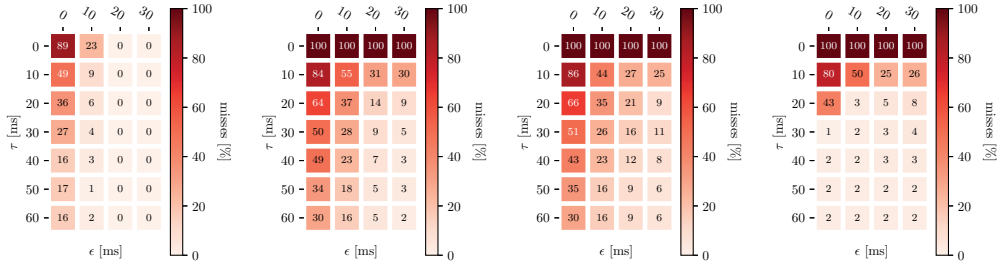
Figure 4 shows the hyper-plane average age of information vs. ratio updates-requests in the ideal communication channel case. We compare Age-Optimal π^*_Δ, Periodic π_P, Grouping Window π_G and Grouping Window with Delay Budget π_D

[1] ω as from Equation 7 with $\epsilon \geq 0$, while $\epsilon = 0$ only for pseudo-code in Algorithm 1, not for Equation 7 and 8.



(a) No delay: Age of Information (b) No delay: Response time (c) With delay: Age of Information (d) With delay: Response time

Fig. 5. System metrics evolution varying τ and ϵ on perfect links and links with random delay for the Grouping Window with Delay Budget schedule π_D. The data points ($\epsilon = 0$) correspond to π_G, the data point ($\tau = 0$, $\epsilon = 0$) to π^*_Δ.



(a) No delay: periodic π_P (b) With delay: periodic π_P (c) With delay: Grouping Window with Delay Budget π_D (d) With delay: Miss Avoidance Grouping Window π_ω

Fig. 6. Percentage of misses evolution varying τ and ϵ for different scheduling policies.

scheduling policies. The star on the bottom-right corner represents the Age-Optimal schedule which provides best possible AoI and maximum ρ, as this strategy schedules an update for every application request. The up most dotted line shows the evolution of the periodic schedule. We clearly see that the smaller the period, the lower the AoI at the cost of higher ρ. Roughly, $\Delta_A \approx T_s/2$. Both π_G and π_D deliver better AoI for the same updates/request ratio as they are designed to meet applications requirements. The applications will be scheduled in between a time window that, in the best case, provides AoI $0\,\text{ms}$ and, in the worst, $40\,\text{ms}$. Δ_A depends on the distribution and the number of applications in the window. In the illustrated example, $\pi_P(51)$, $\pi_G(40)$ and $\pi_D(40, 10)$ for $\rho = 0.26$, π_G and π_D improve Δ_A by 40% and 66% respectively. The reason for π_D to be better than π_G, in terms of AoI, is that the applications are willing to pay some response time ReT for fresher data.

B. Age of Information and Response Time

Figure 5 shows the evolution of the system metrics with several τ and ϵ configurations for the π_D schedule, both on a perfect link and a link affected by delays. Fig. 5a shows the average AoI, Δ_A. Reading the heat map from top to bottom, we evince that the bigger τ the higher Δ_A, intuitively because of the larger grouping window contributing to higher $\Delta(\cdot)$. For fixed τ values, the bigger ϵ the higher the response time ReT, see Fig. 5b. As a result, Δ_A decreases because the applications are willing to wait some time to get fresher information. To be noticed, ReT is significantly smaller than the tolerance ϵ. Reading the matrix from top to bottom, we notice an increment then decrement in ReT. In fact, when $\tau < \epsilon$, the applications are willing to wait time to fetch fresher data, that will come because of the smaller grouping window. As a result, the average response time increases. On the other hand, when $\tau > \epsilon$, updates come more seldom and the applications will not wait: this result in smaller average ReT.

Figure 5c and 5d present the same metrics on a link affected

by delays. The same behavior described for the ideal link applies in this case as well, with the difference that we must account for an additive factor coming from the delays. These contribute to increase Δ_A, especially for big τ and small ϵ. In fact, they affect particularly the applications that are scheduled between the beginning of the grouping window and the extent of the delay itself. To make things even worse, if these applications have no delay budget, they will not be able to compensate for delays, delivering poor AoI and higher chances of misses. The response time is not significantly affected by random delays as we assume an oracle digital twin, see Sec. V-A.

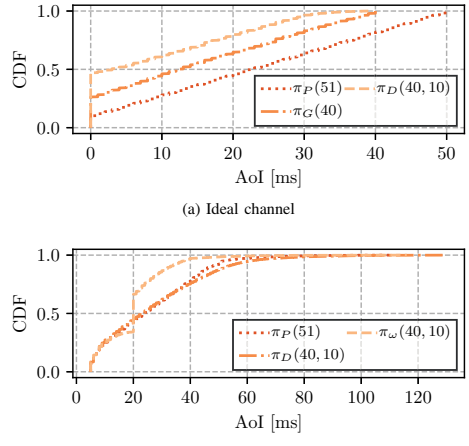
C. Misses

We now consider how π_P, π_D and π_ω are affected in terms of deadline misses. For this particular experiment, we aim at 5% misses and, because of the used Pareto distribution, we obtain $d_M = d_{95} = 30\,\text{ms}$. Fig. 6 provides an overview of the evolution of misses, while varying τ and ϵ for the aforementioned policies. As a remainder, a miss happens when the sensor update is consumed by the application when it is older than τ. We consider first π_P, whose period is calculated as the mean value of the inter-arrival time of the relative Grouping Window schedule used for comparison. To put it in other words, the two schedules will have the same ρ. Fig. 6a shows the performance of the periodic schedule on a perfect link. We clearly see that, without delay budget, the percentage of misses is rather high. On the other hand, if the applications have delay budget, the periodic schedule is able to compensate. When introducing random delays, see Fig. 6b, the periodic scheduling delivers many more misses than in the previous case. Again, if the applications do not account for some delay budget, the performance is very poor.

Fig. 6c shows how the Grouping Window with Delay Budget is affected by delays. The results are in line with the periodic schedule as, for some configuration of (τ, ϵ), it is slightly better and, for others, slightly worse. Nonetheless, both Fig. 6c and 6b exhibit degraded performance. For this reason, we introduced Miss Avoidance Grouping Window π_ω, and the obtained percentage of misses is shown in Fig. 6d. For $\tau \geq 20\,\text{ms}$[2] we notice a significant performance improvement both over π_P and π_D. Surprisingly, and contrarily to all the previous cases, some increasing ϵ deliver slightly higher number of misses. The explanation behind this is rooted in Equation 7. In fact, the bigger ϵ the bigger the grouping window, meaning more applications executions in it. When a random delay is greater than d_{95}, all the applications in the window will experience a deadline miss. That is, increasing ϵ in π_ω may lead to deliver higher number of misses, yet reducing ρ. Nonetheless, such increment of misses is negligible.

D. Age of Information Distribution

Fig. 7 presents the cumulative distribution function (CDF) of AoI on both an ideal channel and a channel affected by delays. The former case is depicted in Fig. 7a. We observe that all the schedules exhibit a linear distribution and the CDFs resemble parallel lines. The main difference between the schedules is the amount of zeros. In fact, zeros are distributed such that $\pi_G \approx 25\%$ and $\pi_D \approx 50\%$, while $\pi_P = 0\%$. The reason behind the higher number of zeros in π_G is that the schedule is optimized for delivering best AoI to the first application of every grouping window. On the other hand, π_D delivers even more zeros because it serves best AoI to *at least* one application in every grouping window. In fact, applications with delay budgets can be postponed such that more applications will read best AoI at the time of the new update. From the plot, we als notice an interesting property of AoI for periodic applications: π_P delivers AoI that is uniformly distributed in $[0, 51]$.



(a) Ideal channel

(b) Channel with random delays

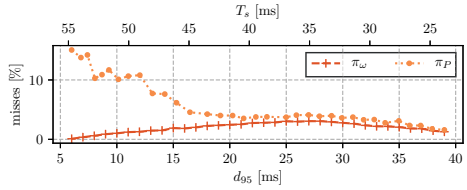
Fig. 7. CDF of average age of information for different scheduling policies.

Fig. 7b shows the CDF of AoI for π_P, π_D and π_ω on a channel affected by delays. π_D and its relative periodic, π_P, have a similar distributions as they approximately share the same ρ. The only difference between the two schedules is that π_P delivers updates regularly while π_D may deliver either a bit earlier or a bit later according to applications schedule. Furthermore, none of the policies implement a mechanism to mitigate delays. On the other hand, π_ω takes into account "safety margins" for avoiding to serve applications with stale data due to delays. As a result, we observe significant improvement on the X-axis between $20\,\text{ms}$ and $60\,\text{ms}$.

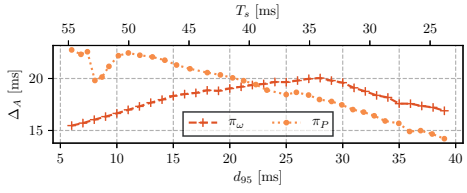
E. A broader picture

It is difficult to provide a complete overview of periodic and Grouping Window scheduling policies. In fact, there are many possible parameters combinations and the results are strongly dependent on the used delay distribution. In this section, we select one of these configurations ($\tau = 40\,\text{ms}, \epsilon = 10\,\text{ms}$) and we study it over a range of delay distributions. Our goal

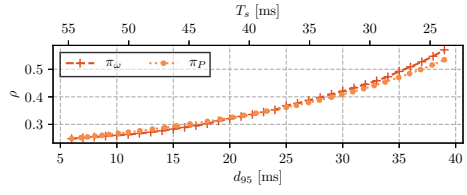
[2]For $\tau < 20$ the used delay distribution would make ω negative but, in that case, we enforce $\omega = 0$.



(a) Misses



(b) Age of Information



(c) Ratio samples-requests

Fig. 8. System metrics evolution for $\tau = 40$ms and $\epsilon = 10$ms on links with random delay. Small d_{95} corresponds to small delays and high d_{95} to long delays.

is to give insights on how to select a suitable scheduling policy on the basis of the delay distribution. To that end, we adapt the shape parameter α of the Pareto delay distribution to vary its 95:th percentile d_{95}. d_{95} is in linear relation to the grouping window ω of π_ω (c.f., equation 7), resulting in smaller grouping windows for higher delay variation.

The figures in Fig. 8 set the periodic and Grouping Window schedules in relation to each other in terms of misses, average system AoI, and overhead. The upper scale of the x-axis corresponds to the average sampling interval T_s resulting from π_ω, which is applied in π_P.

Fig. 8a displays the percentage of misses. Independently from the extent of the delay distribution, the periodic schedule is worse than our proposed schedule. In particular, when $T_s > 30\,\text{ms}$, the number of misses increases dramatically. Furthermore, π_ω is constantly below 3%, achieving the design goal of number of deadline misses mitigation.

Fig. 8b shows the average system AoI Δ_A for the aforementioned settings. We observe that, for the periodic schedule, the greater the period T_s the higher the AoI. On the other hand, the AoI for π_ω increases with the delay variation up to $d_{95} < 28\,\text{ms}$, and decreases after. In fact, the smaller the extent of delay variations the more we get closer to the perfect link scenario, where we already showed that the Grouping Window approach outperforms the periodic. We find particularly interesting that through this plot it is possible to see which schedule is best according to the delay distribution, e.g., when the two curves intersect at $d_{95} = 22\,\text{ms}$ and the AoI of the Grouping Window schedule exceeds the AoI of the periodic schedule.

Finally, in Fig. 8c we compare the ratio updates-requests. We here observe that both π_P and π_ω are similar. The gap between the curves is to be imputed to the approximation performed to make the two schedules comparable (π_ω mean inter-arrival time).

F. Discussion

We proposed π_G and π_D, two scheduling policies designed to deliver better Δ_A than the periodic schedule π_P. The proposed policies deliver better Δ_A when applied to an ideal communication channel without delays. Nonetheless, when introducing delays in the communication channel, the proposed schedules are not better than the periodic in terms of misses. For this reason, we introduced π_ω.

The Miss Avoidance Grouping Window, π_ω, is a scheduling policy designed to impose an upper bound on the number of misses induced by communication delays. We show that π_ω always delivers fewer misses than π_P. Nonetheless, when the delay variations are big, the periodic policy is able to deliver better Δ_A than π_ω. This is due to the fact that π_ω imposes a safety margin that results in sending updates unnecessarily early.

We now discuss some implementation challenges. First, the proposed scheduling policies are sensitive to the notion of time. To some extent, the safety margin introduced to handle random delays can also compensate for timing inaccuracies. Furthermore, as the number of applications grows, the schedule produced by the edge device may be very long and not suitable to be stored in resource constrained devices, e.g., sensors. In this case, the edge should fragment such schedule, push partial information to the sensor while keeping track of the progress and renew the schedule when needed, for every sensor. However, such strategy may be merged together with schedule updates sometimes required by new applications joining the system.

VII. Conclusion

We studied the problem of timely dissemination of sensor data to applications with freshness requirements by means of a digital twin. We minimize direct access to the possibly battery powered physical devices yet improving Age of Information as a data freshness metric. The proposed Age of Information-aware scheduling policies allow the physical devices to push sensor updates to the digital twin by grouping application requests based on freshness criterion, thereby reducing the number of sensor transmissions. Scheduling updates in a network with communication delays is non-trivial. In fact, random delays affect the Age of Information and are likely to result in deadline misses, if the requirements are too stringent. By estimating the random delay component, we introduce some

safety margin in the schedule to keep the misses below a target level. Delaying replies to application requests in the favor of awaiting fresh data has proven to be an efficient measure. We numerically evaluate the proposed policies against a simple yet widely used periodic scheduling and demonstrate that our schedule, even with high delay variation, outperforms the periodic schedule in terms of misses, at the cost of marginally higher Age of Information.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their constructive comments. We are grateful to Bengt Ahlgren and Anders Lindgren from RISE for discussions. This work was supported by the Swedish Foundation for Strategic Research under the project "Future Factories in the Cloud (FiC)" with grant number GMT14-0032.

REFERENCES

[1] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, June 2011, pp. 350–358.

[2] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2731–2735.

[3] K. Chen and L. Huang, "Age-of-information in the presence of error," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 2579–2583.

[4] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and value of information: Non-linear age case," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 326–330.

[5] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1360–1374, March 2016.

[6] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, April 2016.

[7] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 2574–2578.

[8] R. D. Yates and S. Kaul, "Real-time status updating: Multiple sources," in *2012 IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 2666–2670.

[9] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 3008–3012.

[10] B. T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu, "Age of information under energy replenishment constraints," in *2015 Information Theory and Applications Workshop (ITA)*, Feb 2015, pp. 25–31.

[11] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[12] T. Bacinoglu and E. Uysal-Biyikoglu, "Scheduling status updates to minimize age of information with an energy harvesting sensor," in *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, 06 2017, pp. 1122–1126.

[13] X. Wu, J. Yang, and J. Wu, "Optimal status update for age of information minimization with an energy harvesting source," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 1, pp. 193–204, March 2018.

[14] S. Feng and J. Yang, "Optimal status updating for an energy harvesting sensor with a noisy channel," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 348–353, 2018.

[15] B. Li, A. Eryilmaz, and R. Srikant, "On the universality of age-based scheduling in wireless networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 1302–1310.

[16] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2016, pp. 844–851.

[17] Q. He, D. Yuan, and A. Ephremides, "Optimizing freshness of information: On minimum age link scheduling in wireless systems," in *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2016, pp. 1–8.

[18] Y. P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 561–565.

[19] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '18. New York, NY, USA: ACM, 2018, pp. 191–200. [Online]. Available: http://doi.acm.org/10.1145/3209582.3209602

[20] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[21] S. Ioannidis, A. Chaintreau, and L. Massoulie, "Optimal and scalable distribution of content updates over a mobile social network," in *IEEE INFOCOM 2009*, April 2009, pp. 1422–1430.

[22] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[23] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[24] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[25] K. M. Alam and A. E. Saddik, "C2ps: A digital twin architecture reference model for the cloud-based cyber-physical systems," *IEEE Access*, vol. 5, pp. 2050–2062, 2017.

[26] W. Zhang and J. He, "Modeling End-to-End Delay Using Pareto Distribution," in *International Conference on Internet Monitoring and Protection (ICIMP)*, 2007.

Paper III

Pruning Edge Research with Latency Shears

Nitinder Mohan
Technical University of Munich
mohan@in.tum.de

Lorenzo Corneo
Uppsala Universitet
lorenzo.corneo@it.uu.se

Aleksandr Zavodovski
University of Helsinki
aleksandr.zavodovski@helsinki.fi

Suzan Bayhan
University of Twente
s.bayhan@utwente.nl

Walter Wong
University of Helsinki
walter.wong@helsinki.fi

Jussi Kangasharju
University of Helsinki
jussi.kangasharju@helsinki.fi

ABSTRACT

Edge computing has gained attention from both academia and industry by pursuing two significant challenges: 1) moving latency critical services closer to the users, 2) saving network bandwidth by aggregating large flows before sending them to the cloud. While the rationale appeared sound at its inception almost a decade ago, several current trends are impacting it. Clouds have spread geographically reducing end-user latency, mobile phones' computing capabilities are improving, and network bandwidth at the core keeps increasing. In this paper, we scrutinize edge computing, examining its outlook and future in the context of these trends. We perform extensive client-to-cloud measurements using RIPE Atlas, and show that latency reduction as motivation for edge is not as persuasive as once believed; for most applications the cloud is already "close enough" for majority of the world's population. This implies that edge computing may only be applicable for certain application niches, as opposed to a general-purpose solution.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Cloud computing**.

KEYWORDS

Edge computing; Cloud computing; Internet measurements; Cloud reachability

ACM Reference Format:
Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. 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, 8 pages. https://doi.org/10.1145/3422604.3425943

1 INTRODUCTION

Edge computing has emerged as a new, compellingly sounding solution for improving and enabling many network applications. One selling point of edge is improving latency by moving services closer to end-users and pre-processing at the "edge" to save the network (and cloud) from being overwhelmed by unforeseen amounts of data [17]. This enables sophisticated applications, e.g., augmented and virtual reality [23, 42], robotic control [14, 21, 45], smart homes/cities [4], etc. Many concrete scenarios have been developed [11, 28] and industrial standardization initiatives, e.g., multi-access edge computing are actively promoted by telcos [24, 67].

However, since edge's inception around a decade ago, several current trends have emerged which *may* impact its utility. *First*, cloud infrastructure is spreading geographically by installing new datacenters. For example, since 2010, Amazon has expanded its cloud network from 3 to 16 countries. Furthermore, cloud providers are establishing (and incorporating) specialized facilities to tackle edge needs, e.g., CloudFront [2]. *Second*, modern smartphones come equipped with considerable processing power, including specialized chipsets, enabling them to process complex tasks, such as AR. On the other hand, last-mile access being the latency bottleneck over (cellular) wireless remains true [12, 31]. While new technologies, such as 5G, show promise, initial tests report their performance to be deficient in practice [49, 71]. *Third*, offloading [19, 58, 59], and cyber foraging [8, 30], have become a reality with services like Siri or Cortana. Products with tight latency constraints, e.g., cloud-based gaming, are already on the market [3, 29, 44], implying improved cloud access latencies. Recent study from Facebook reveals that most users can reach their services in the cloud within 40 ms [60].

We believe these trends necessiate *pruning* popular assumptions driving edge computing research and *identifying* more promising future directions. We achieve this by examining the latency for connecting to cloud globally. Our contributions are as follows.

(1) We conduct large-scale measurements over RIPE Atlas analyzing user reachability to datacenters owned and operated by seven cloud providers. Our measurements targeted 101 datacenters in 21 countries and lasted several months. Only [36] has conducted a similar study to ours but it is limited to single cloud provider; the most recent multi-cloud measurement is a decade old [40].

(2) We take a critical look at edge computing and its future potential, by analyzing latency and bandwidth thresholds of several applications reputedly enabled by edge computing. Extrapolating our measurement results, we find that, contrary to popular belief, the effectiveness of edge computing is limited to a few applications, such as traffic monitoring, gaming, etc., which, incidentally, are not the primary drivers of edge hype. Other applications can be either supported by current cloud infrastructure (smart home, wearables) or *will* require onboard processing for optimal operation.

Edge computing is still in flux. Some [27] see edge taking over from the cloud; others see a combination [55]. Peterson et al. [52] see edge and the democratization it offers as a cure for Internet ossification. Some argue for wide-spread in-network computation [57],

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

ACM ISBN 978-1-4503-8145-1/20/11.
https://doi.org/10.1145/3422604.3425943

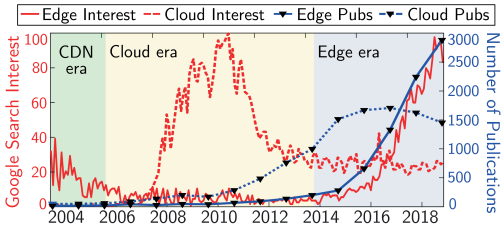


Figure 1: The popularity (in red) and publications (in blue) of keywords "edge computing" (in solid line) and "cloud computing" (in dashed line) in Google web searches and Google scholar respectively.

blurring the borders of cloud and edge [32, 75]. Our focus, in this paper, is on a *general-purpose edge deployed by telcos/ISPs* for a wide range of applications [47]. While we show this path to have substantial hurdles, there may be better-suited scenarios for edge. We return to these at the end of the paper.

2 A RETROSPECTIVE ON EDGE

Figure 1 captures the zeitgeist of "edge computing" over the past fifteen years. It compares the frequency of Google web searches[1] and scientific publications[2] for "edge computing" and "cloud computing" from 2004 to 2019. Resultingly, three eras can be distinguished: **content delivery networks (CDN)**, **cloud**, and **edge**.

The term *edge* emerged when ***CDNs*** started to deploy edge servers near their clients [20]. They acted as caches of content, speeding up content delivery and reducing bandwidth usage. At the same time, centralized, large-scale datacenter deployments emerged, heralding the ***Cloud era***. Cloud was a success as the type and volume of application's resources could be elastically adjusted to meet the current demand on-the-fly. Application developers could also take advantage of a flexible *"pay-as-you-go"* model for resource utilization.

Cloudlets [59] in 2009 started the ***Edge era*** and similar concepts, such as *fog computing* [9]. Back then, the cloud was limited to a few datacenters and unable to address the stringent latency and data transport requirements of new use cases, such as the Internet-of-Things (IoT). Therefore, the research community, including industry, jumped at the opportunity to decouple network latency from the computation time, and devised "edge computing". Many edge architectures have been proposed [46, 48], including exploiting last-mile access points [13], crowdsourcing [26], and using IoT sensors [53]. We will next take a systematic look at various applications driving the hype on edge computing and analyze their requirements.

3 DRIVERS OF THE EDGE HYPE

We capture applications used to motivate edge computing in Figure 2. The y-axis is the required latency scale, ranging from a few milliseconds (ms) to an hour (hr). The x-axis shows the amount of

[1]Results obtained from https://trends.google.com/.

[2]Data was collected by a custom web crawler for Google Scholar, based on an open source implementation [38].

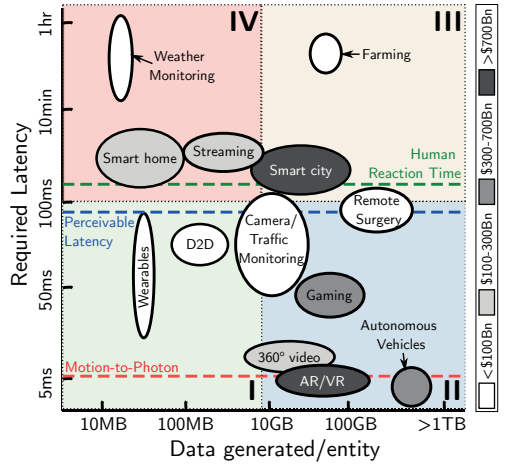
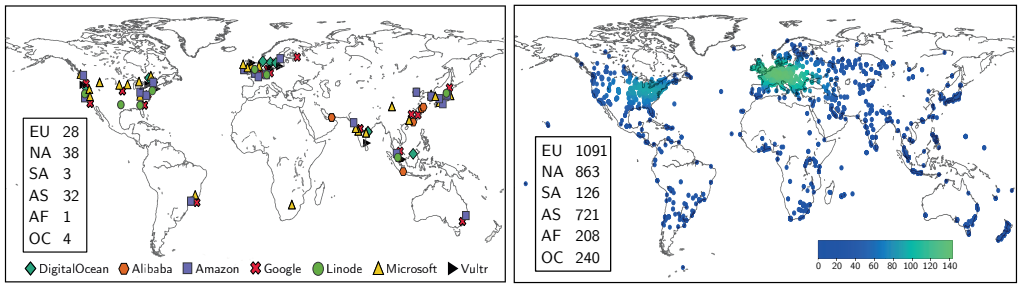


Figure 2: Driving edge applications represented as ellipses. The orientation and width signify strictness in bandwidth and latency requirements. Color denotes the expected market share by the year 2025.

data an entity of each application generates, e.g., a camera, which naturally correlates with the network bandwidth requirements. We estimate application requirements by relying on theoretical analysis and preliminary implementations from previously published results [7, 37, 42, 54, 64]. Each application is represented as an ellipse to overcompensate for any estimation errors. The form and orientation of the ellipse represent the application's *strictness* towards latency/bandwidth constraints. The ellipse's color denotes application's expected market share by 2025 in US dollars (data is from [63]). Majority applications in Figure 2 are human-centric – taking inputs and providing feedback to users, e.g. gaming. The QoE of such applications is governed by strict latency thresholds as human senses require, which we also draw in the figure.

(1) Motion-to-Photon (MTP) is the delay between user input and its effect to be reflected on a display screen. MTP is guided by the human vestibular system, which requires sensory inputs and interactions to be in complete sync; failure of which results in motion sickness and dizziness. Maintaining latency below MTP, i.e., $\lesssim$ 20 ms, is critical for immersive applications, such as AR/VR, 360° streaming, etc. [43]. Of this, $\approx$ 13 ms is taken up by the display technology due to refresh rate, pixel switching, etc. which leaves a budget of $\approx$ 7 ms for computing and rendering (including RTT to server) [16]. Studies by NASA concludes that certain HUD systems may require the compute part of MTP to be as low as 2.5 ms [7].

(2) Perceivable Latency (PL) is the threshold when the delay between user input and visual feedback becomes large enough to be detected by the human eye [54]. PL threshold plays a vital role in the QoE of applications where the user interaction with the system is fully



(a) Distribution of cloud regions from seven major cloud providers.

(b) Distribution of 3200+ RIPE Atlas probes.

Figure 3: Cloud regions with compute DCs in (a) represent our targets, and probes in (b) are the vantage points for our study.

or semi passive, e.g., video streaming (stuttering), gaming (input lags), etc. It is roughly estimated to be 100 ms.

(3) Human Reaction Time (HRT) is the delay between the presentation of a stimulus and the associated motor response by a human. While HRT is highly dependent on the individual (and can be improved by training), its value is reported to be ≈ 250 ms [73]. Applications that require active human engagement, e.g. remote surgery, teleoperated vehicles etc., must operate within HRT.

Considering the similarities in operational thresholds, we group the emerging applications by quadrants.

Quadrant I - *Low Latency & Low Bandwidth*: The bottom-left (green) quadrant represents applications that produce only a small volume of data but impose strict latency constraints for optimal operation. Typical examples include wearables, health monitoring, and other individual-focused applications. The core aim of applications in *Q1* is to perform "naturally", i.e., to operate within the PL threshold. Hence, they can benefit from the low latency computation promised by the edge.

Quadrant II - *Low Latency & High Bandwidth*: The blue quadrant at bottom-right encompasses applications that generate large data volumes *and* impose strict latency constraints, e.g. autonomous vehicles, AR/VR, cloud gaming, etc. Edge computing is considered to be the key enabler for applications in *Q2*, as additional latencies to compute at traditional cloud and bandwidth strain on backhaul networks to transport generated data may break the "immersiveness" of end-users [37]. As most applications in this quadrant are expected to garner large market shares, these are popularly heralded as the driving force behind edge computing.

Quadrant III - *High Latency & High Bandwidth*: The top-right quadrant (yellow) is composed of applications that generate large volumes of data but with "somewhat" relaxed timing constraints. Take, for example, a smart city that implies automatic updates on buses timetables, smart parking meters, and overall maintenance with control mechanisms. The demand *Q3* applications place on edge computing is usually limited to data aggregation and preprocessing to reduce network bandwidth load.

Quadrant IV - *High Latency & Low Bandwidth*: The final quadrant in red, located top-left, comprises of applications that neither generate data of large volumes nor require strict latency for operation, e.g. smart homes, weather monitoring, etc. While such applications can leverage the existence of edge computing, they do not offer compelling reasons for deploying edge servers.

4 AT THE EDGE OF THE CLOUD

One key driver of edge computing is its claimed ability to provide services at lower latencies than the cloud. While this claim was valid at the emergence of edge computing (circa 2009) due to sparse cloud deployment [40] and higher latencies in the core network than at the last-mile [39]; the world (and cloud) has changed in the past decade. For instance, Amazon's cloud has increased from 3 to 22 datacenter locations [1], and wide area latencies to Google's CDN have decreased from 100 ms to 10-25 ms [61]. On the other hand, edge computing is still in its infancy with no (wide-area) deployment to date, so the claims seem more speculative than real.

We re-evaluate the latency-centered claims of edge computing via extensive global wide-area measurements to datacenters of major cloud providers. We aim to understand if the cloud access latency is still too high in satiating the requirements of emerging applications, or are the clouds already "close enough".

4.1 Measurement Methodology

End-Points. We chose 101 cloud regions with compute datacenters (e.g. ec2) from seven cloud providers, Amazon, Google, Microsoft Azure, Digital Ocean, Linode, Alibaba, and Vultr, as end-points, shown in Figure 3a, and established a VM in every selected location. The chosen operators are widely used and provide global coverage with distinct network infrastructure. Some, e.g. Amazon, Google etc. have installed *private*, large bandwidth, low latency network backbones with wide-scale ISP peering, while others, e.g. Linode, largely rely on the *public* Internet for connectivity.

Vantage Points. We used 3200+ RIPE Atlas probes [62] distributed in 166 countries as vantage points for our measurements (shown in Figure 3b). RIPE Atlas is a global Internet measurement platform that is widely used for reachability, connectivity, and performance studies. Atlas probes are installed in varying network environments

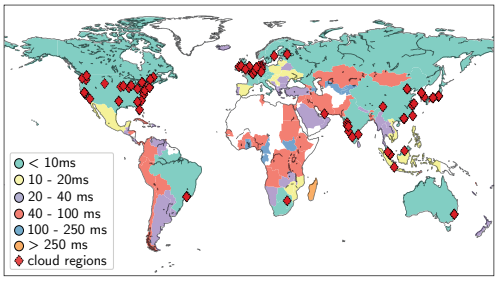


Figure 4: Minimum latency to nearest datacenter globally.

(core, access, or home), allowing us to analyze the user reachability to the cloud globally. We filter out all the probes that are clearly installed in privileged locations (e.g., datacenters, cloud network) from our measurements using their user-defined tags [6].

Experiment. We measured end-to-end latencies between users (Atlas probes) and cloud datacenters within the same continent via `ping` every three hours. For probes in continents with low datacenter density, e.g., Africa and South America, we also measured latencies to datacenters in adjacent continents, i.e., Europe and North America. Our measurements are ongoing since *September 2019*, and the results in this work are drawn from *nine months* of data collection. Overall, our dataset includes 3.2 million datapoints spanning several GBs and is available for public use [18].

4.2 Proximity to the Cloud

What is the least latency with which countries can access the nearest datacenter? The question allows us to analyze the spread of cloud across the globe in terms of latency. We extract the minimum `ping` latency observed by the best-performing probe for every country to any cloud datacenter. Figure 4 shows the map of latency distribution per country. The results show that 32 countries can access the cloud with RTTs less than 10 ms, and another 21 countries with RTTs between 10 to 20 ms (MTP threshold). Our findings become more intuitive upon correlating geographical latencies to locations of targeted datacenters (red diamonds) in Figure 4. Countries with cloud access latency less than 10 ms typically have one or more local datacenters, and those with latencies less than 20 ms either share borders or have direct fiber connectivity [68] to the country housing a datacenter. In fact, all *but* 16 countries (mostly in Africa) can access the cloud within PL threshold bounds (100 ms).

While the above shows only the best probe in every country, Figure 5 plots the CDF of the minimum latency observed from *every* probe in our dataset to *any* datacenter, grouped by continents. The result includes probes without a stable Internet connection, or with wireless connectivity. Despite this, the results support the findings in Figure 4. Around 80% probes in Europe and North America – ≈50% of our total probes – can access a cloud datacenter within MTP (20 ms). Probes in Oceania follow similar performance pattern as almost all of them can access the cloud within 50 ms RTT. Surprisingly, despite the low availability of cloud regions and substandard network deployment, ≈75% probes in Africa and Latin America achieve less than 100 ms cloud access latency and meet PL thresholds. While the results in this section are "optimistic" – in that they show the *minimum* latency – they also indicate that the cloud potentially can support latency requirements for applications driving edge computing.

4.3 Where is the Delay?

Insufficient Infrastructure Deployment. Our results above focused on best-case scenarios to illustrate the *potential* reach of the cloud. We now turn to our entire latency dataset to shed light on the *reality* of the cloud. Figure 6 shows the latency distribution of all measurements grouped by continent. Probes in North America, Europe, and Oceania exhibit excellent cloud reachability, with *more than 75%* of the probes achieving RTT below the PL threshold. The top 25% probes in NA and EU can even support MTP threshold required by edge-compelling applications, e.g. AR/VR, autonomous vehicles, etc. The reason for this exceptional performance (also hinted in §4.2) is the concentrated efforts of cloud providers to deploy datacenters in these continents. Note that the long tail of latency distribution for EU is largely missing from NA. On deeper analysis, we found that the primary contributors to the tail are probes in eastern EU and countries without local or neighboring datacenters, in line with our assessment from Figure 4.

We now turn our focus on the remaining continents, i.e., Latin America, Asia, and Africa. Cloud reachability from within these continents is quite poor, and only a fraction of probes can satisfy the PL threshold. Probes in Asia show much diverse latencies primarily due to scattered datacenter deployment favoring certain countries, like China and India. Unsurprisingly, the worst performance is in Africa as it is severely under-served, both in cloud presence (only one operating region) and network infrastructure [15].

Nature of last-mile access. Many studies analyzing the performance of wireless access have been conducted in the past. Be it WiFi in home networks [66] or LTE in public spaces [50], the consensus of last-mile being the bottleneck is well established. Reasons for lack of wireless performance can be many, from packet drops due to contention, to network bufferbloating because of handovers [35]. As most applications in Figure 2 rely on wireless, we also analyze its impact as access medium to the cloud.

We leverage RIPE Atlas user-provided tags [6] to filter probes which indicate the type of access link, e.g. *ethernet, broadband* for wired and *lte, wifi, wlan* for probes connected to network through wireless links. We further filter probes deployed in similar regions in both sets and verify that their baseline latency is in line with their country's average. Figure 7 compares the latencies observed by both sets throughout our measurement period. We find that probes tagged with wireless keywords perform consistently worse than their wired counterparts – taking ≈2.5× longer to access the nearest cloud region. Our result is in line with previous studies showing that users can experience 10-40 ms of added latency while using wireless as last-mile [65, 66]. While these results might improve in future with solutions such as 5G promising much shorter wireless latencies, however, the technology is still in its nascent stages and these promises are waiting to be delivered [71].

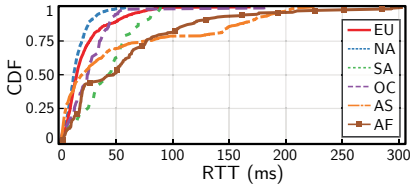


Figure 5: CDF of minimum RTT of all probes to nearest datacenter by continent.

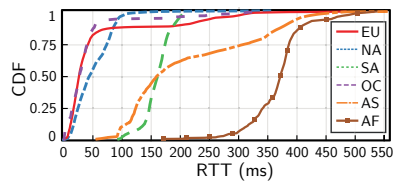


Figure 6: CDF of all ping measurements from all probes to their closest datacenter.

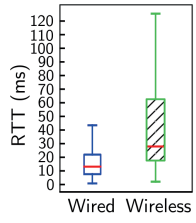


Figure 7: Wired vs. wireless access RTT.

5 DISCUSSION

Revisiting Edge Applications. We reconsider Figure 2 with our updated knowledge on real-world latencies from our measurements. As per §4.3 and previous measurement studies, current wireless technologies do not support access link latencies below 10 ms. While new wireless standards *promise* to improve the situation, e.g., 10 Gbps speeds with WiFi-6 [22] and 1 ms latency with 5G [34], the reality may differ from claims. For example, at its inception in 2011, LTE promised access latencies below 10 ms while the user is near the base station [25]. However, recent measurements show that the standard commonly experiences delays lasting several seconds due to queue build-ups [35] and handovers [72]. Recent investigations report performance of preliminary deployment of 5G in the real-world to be sub-optimal [49, 71]. Likewise, Hadzic et al. [31] and Cartas et al. [12] find that latency gains for accessing edge server colocated with an LTE basestation is minimal compared to accessing a datacenter located ≈ 1000 km away. While the "true" gains of 5G are yet to be seen, considering supporting strict MTP thresholds, even with edge servers located at basestations, seems uncertain. From §4.2, we can conclude that for most of Europe and North America (and majority of the world in best case), cloud latencies are low enough to support applications operating under perceivable latency. On the contrary, due to lack of network infrastructure, some countries (in Asia & Africa) see cloud access latencies of 150–200 ms, making perceivable latency unachievable but HRT-based applications feasible by the cloud.

Figure 8 recaps the edge applications and their network requirements but now adds latency (red) and bandwidth (blue) "reality" boundaries, as shaded regions (based on the results in §4). The lower bound on latency is ≈10 ms, i.e., the current state of wireless access latency. The upper bound is the human reaction time – as this is supported by the cloud almost globally. For bandwidth, edge is most useful for applications generating enough data to congest the network. Specifically, benefits from the edge are greatest close to the users, and decrease with increasing distance. Contrarily, it is also well-established that the primary bandwidth bottleneck is usually the last-mile [66]. While last-mile bandwidth congestion also depends on contention and competition, based on previous studies [35], we estimate 1GB/entity data generation to be a fitting threshold for edge's bandwidth aggregation gains.

The overlap is the "*feasibility zone*" (FZ) of edge computing. Applications in this zone, e.g., traffic camera monitoring, cloud gaming,

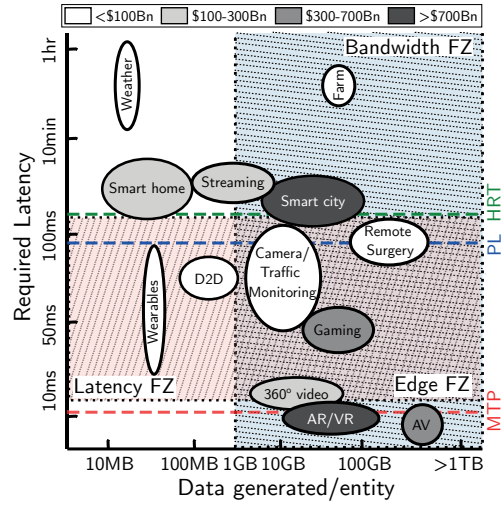


Figure 8: Edge applications but with edge computing feasibility zones (FZ). The red shaded area represents potential latency gains, and the blue shaded region is the bandwidth gain zone for utilizing edge.

etc., clearly benefit from a wide deployment of edge as they impose both latency and bandwidth constraints. Surprisingly, the primary drivers of edge computing research (the ones with the most hype) *do not* fall in this zone. For some, it is due to low bandwidth requirements (e.g., wearables), and for others, it is either too stringent (e.g., autonomous vehicles) or too relaxed (e.g., smart cities) latency constraints. Interestingly, the predicted market share of applications within the edge FZ pales compared to those for which edge does not provide much benefit. Further, many applications in the edge FZ can be supported by a wider deployment of cloud/network infrastructure, especially in Asia, Latin America, and Africa.

Other Considerations. We recognize that our critique above may not fully encompass the utility of edge computing due to possible limitations in our methodology, emerging application diversity or

other complications that we may not have considered. For example, while we place applications in edge FZ based on their latency and bandwidth requirements, factors – like privacy – may push other neighboring applications into the FZ envelope. We identify several such factors that may impact our findings and require further research investigation.

Network vs. application latency. It may be argued that our viewpoint does not include additional processing delays imposed by applications as we derive network latency from `ping`. However, a recent study from Facebook report similar results as ours and shows that clients rarely observe latencies above 40 ms while accessing their services hosted in the cloud globally [60]. Furthermore, we plan to extend our measurements to include TCP-based probing techniques [41] that may better reflect behavior of application traffic inbound cloud networks.

Computing power: Our discussion in this paper did not consider the differences in computation power between the cloud and edge servers. The more pervasive deployment edge needs, the lower the likely processing capabilities of individual edge servers become. It is thus quite possible that despite extensive edge deployment, faster processing and availability of specialized hardware (like GPUs) offered by the cloud may far exceed the network latency gains from deploying applications at the edge [12].

Economies of scale: One decisive advantage cloud computing is economies of scale, which edge is unlikely to meet. For cloud, aggregating a large number of servers in a single location achieves substantial savings on building, maintaining, and securing the infrastructure. For edge, marked gains in latency are possible only via a wide and expensive deployment. While last-mile ISPs are best placed to exploit this, recently, cloud providers have begun to utilize the ISP edge [10, 70] and CDN infrastructure [2], further bringing cloud closer to users. However, as the cloud footprint expands to support lower latency requirements of emerging applications, the cloud infrastructure cost will also increase [33].

6 FUTURE RESEARCH DIRECTIONS

With our global measurements, we showed that one of the compelling motivators behind the edge – *reduction in latency* – has lost much of its importance since the inception of the field almost a decade ago. Through the course of this study, we found that latency is only one piece in the puzzle and other considerations may serve as more convincing drivers for edge. To conclude, we outline some more promising directions for future research in edge computing.

Plausible deployments. General-purpose edge yields little benefits in well-connected areas, but in developing regions, gains are more significant, making edge deployment more compelling. Efforts, therefore, should instead focus on those regions for deployment. Noghabi et al. [51] lists some application-specific deployments where edge may offer benefits even in developed regions, e.g. handling video feeds from traffic cameras. Such deployments are typically purpose-built to support an organization's workload and may emerge as the preferred solution in lieu of generic telco-hosted edge. However, the race towards deploying an edge infrastructure can be viewed as tussle between ISPs and cloud providers, both competing for bigger share in "compute" market, which may sway plausible deployments favoring one network type over the other. In this case, research related to tradeoffs in placement and utilization of processing capacity may yield interesting insights.

Privacy has been brought up as an advantage of edge, as it obviates the need to send (sensitive) data to a central cloud. Encryption alone may not be sufficient to hide all details [5, 56]. As concerns for monitoring [69] and data collection mount, processing local data locally and not sending it to the cloud oligopoly may become more attractive. We see the potential for edge computing to address these concerns, especially for (i) applications with a geographically limited scope of interest and (ii) deployments offered by multiple local providers [74] which are safeguarded by rules of the land.

Trust and security. Cloud operators invest heavily in infrastructure security, but how would the situation translate for a widespread deployment of edge in remote locations? Could the same be assumed of the (possibly myriad of) edge operators? What guarantees would an application provider have in these cases? In the cloud the terms-of-operation agreement is between service and a cloud provider and, in case of problems, litigation can be used. Translating this to an edge with multiple participating (somewhat transparent) entities requires much additional work.

7 CONCLUSION

In this paper, we investigated the rationale behind edge computing in light of recent trends in cloud computing. Still much favored by research and industry, we showed that original motivations for edge computing are weak in today's Internet. We performed an extensive measurement study lasting several months, using probes from RIPE Atlas platform in 166 countries, to measure the proximity towards deployment of modern cloud providers. We found that in well-connected areas, like Europe or North America, the cloud is able to satisfy almost all application requirements that have been envisioned for edge. The remaining ones may continue to remain infeasible for immediately foreseeable future, as they depend on the last-mile wireless access latency. While new technologies, like 5G, promise to improve the last-mile connectivity, related studies measuring its performance over initial deployment show sub-optimal results and full-fledged roll-out of 5G will take several years to complete. While there may be other, non-technical drivers for edge computing, our results clearly showed that from a performance point of view, the potential benefits of edge computing remain small. Only in less-connected areas, such as Africa, Latin America, or parts of Asia have we discovered larger benefits from edge computing. *In conclusion*, we believe that the research in edge computing should shy away from latency-centric views and instead focus its efforts in problem areas that are unresolved and can truly benefit from the attention.

ACKNOWLEDGEMENT

We would like to acknowledge RIPE Atlas team for providing us access to their platform and supporting our measurements with increased quota limits. We also appreciate the valuable feedback and insights from Prof. Dr. Jörg Ott for shaping this paper. This work was supported by the Academy of Finland in the BCDC (314167), AIDA (317086), WMD (313477) projects and from the Swedish Foundation for Strategic Research with grant number GMT-14-0032 (Future Factories in the Cloud).

REFERENCES

[1] Amazon. AWS Global Infrastructure. https://aws.amazon.com/about-aws/global-infrastructure/.

[2] Amazon. CloudFront. https://aws.amazon.com/cloudfront/, 2020.

[3] Amazon. Project Luna, 2020. https://www.amazon.com/luna/landing-page.

[4] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *computer*, 50(10):58–67, 2017.

[5] N. Apthorpe, D. Reisman, and N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *CoRR*, 2017.

[6] R. Atlas. Probe tags. "https://atlas.ripe.net/docs/probe-tags/", 2020.

[7] R. E. Bailey, J. J. Arthur, and S. P. Williams. Latency requirements for head-worn display s/evs applications. In *Enhanced and Synthetic Vision 2004*. International Society for Optics and Photonics, 2004.

[8] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang. The case for cyber foraging. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pages 87–92. ACM, 2002.

[9] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*, pages 169–186. Springer, 2014.

[10] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig. Open connect everywhere: A glimpse at the internet ecosystem through the lens of the NetFlix CDN. *ACM SIGCOMM Computer Communication Review*, 48(1):28–34, 2018.

[11] Y. Cao, S. Chen, P. Hou, and D. Brown. Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 2–11. IEEE, 2015.

[12] A. Cartas, M. Kocour, A. Raman, I. Leontiadis, J. Luque, N. Sastry, J. Nuñez-Martinez, D. Perino, and C. Segura. A reality check on inference at mobile networks edge. In *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*. ACM, 2019.

[13] C. Chang, S. N. Srirama, and R. Buyya. Indie fog: An efficient fog-computing infrastructure for the internet of things. *Computer*, 50(9):92–98, 2017.

[14] Y. Chen, Q. Feng, and W. Shi. An industrial robot system based on edge computing: An early experience. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA, 2018. USENIX Association.

[15] M. Chetty, S. Sundaresan, S. Muckaden, N. Feamster, and E. Calandro. Measuring broadband performance in south africa. In *Proceedings of the 4th Annual Symposium on Computing for Development*, ACM DEV-4 '13, New York, NY, USA, 2013. Association for Computing Machinery.

[16] S.-W. Choi, S. Lee, M.-W. Seo, and S.-J. Kang. Time sequential motion-to-photon latency measurement system for virtual reality head-mounted displays. *Electronics*, 7(9):171, 2018.

[17] CISCO. Cisco visual networking index: Forecast and methodology, 2016-2021 (whitepaper). 2017.

[18] L. Corneo, N. Mohan, A. Zavodovski, S. Bayhan, W. Wong, and J. Kangasharju. Pruning Edge Research with Latency Shears: Dataset. "https://mediatum.ub.tum.de/1574595", 2020.

[19] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[20] A. Davis, J. Parikh, and W. E. Weihl. Edgecomputing: Extending enterprise applications to the edge of the internet. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, WWW Alt. '04, pages 180–187, New York, NY, USA, 2004. ACM.

[21] S. Dey and A. Mukherjee. Robotic slam: a review from fog computing and mobile edge computing perspective. pages 153–158, 11 2016.

[22] Digital Trends. How Wi-Fi 6 will transform connectivity in your home, at the office, and beyond. "https://www.digitaltrends.com/computing/wi-fi-6-transform-pc-cisco/", 2019.

[23] M. Erol-Kantarci and S. Sukhmani. Caching and computing at the edge for mobile augmented reality and virtual reality (ar/vr) in 5g. In *Ad Hoc Networks*, pages 169–177. Springer, 2018.

[24] ETSI. Multi-access Edge Computing (MEC). https://www.etsi.org/technologies/multi-access-edge-computing, 2019.

[25] ETSI Technical Report. Feasibility study for Further Advancements for E-UTRA. "https://www.etsi.org/deliver/etsi_tr/136900_136999/136912/09.03.00_60/tr_136912v090300p.pdf", 2011.

[26] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[27] Gartner Maverick Research. The Edge Will Eat the Cloud. https://blogs.gartner.com/thomas_bittman/2017/03/06/the-edge-will-eat-the-cloud/, 2017.

[28] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 356–363. IEEE, 2015.

[29] Google. Stadia. https://stadia.dev/, 2019.

[30] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 153–166. ACM, 2013.

[31] I. Hadžić, Y. Abe, and H. C. Woithe. Edge computing in the epc: A reality check. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, SEC '17, pages 13:1–13:10, New York, NY, USA, 2017. ACM.

[32] D. Haja, M. Szabo, M. Szalay, A. Nagy, A. Kern, L. Toka, and B. Sonkoly. How to orchestrate a distributed openstack. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 293–298. IEEE, 2018.

[33] S. Hasan, S. Gorinsky, C. Dovrolis, and R. K. Sitaraman. Trade-offs in optimizing the cache deployments of cdns. In *IEEE INFOCOM 2014-IEEE conference on computer communications*, pages 460–468. IEEE, 2014.

[34] International Telecommunication Union. Minimum requirements related to technical performance for IMT-2020 radio interface(s). https://www.itu.int/pub/R-REP-M.2410-2017, 2017.

[35] H. Jiang, Y. Wang, K. Lee, and I. Rhee. Tackling bufferbloat in 3g/4g networks. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 2012.

[36] Y. Jin, S. Renganathan, G. Ananthanarayanan, J. Jiang, V. N. Padmanabhan, M. Schroder, M. Calder, and A. Krishnamurthy. Zooming in on wide-area latencies to a global cloud provider. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 104–116, New York, NY, USA, 2019. Association for Computing Machinery.

[37] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui. A measurement study on achieving imperceptible latency in mobile cloud gaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, pages 88–99, New York, NY, USA, 2017. ACM.

[38] C. Kreibich. Parser for Google Scholar. https://github.com/ckreibich/scholar.py, 2017.

[39] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize cdn performance. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 190–201, 2009.

[40] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: Comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 1–14, New York, NY, USA, 2010. Association for Computing Machinery.

[41] Linux Manpage. ITCP Traceroute. "https://linux.die.net/man/1/tcptraceroute", 2020.

[42] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva. Vr is on the edge: How to deliver 360 videos in mobile networks. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 30–35. ACM, 2017.

[43] K. Mania, B. D. Adelstein, S. R. Ellis, and M. I. Hill. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 39–47. ACM, 2004.

[44] Microsoft. Xbox Project xCloud, 2019. https://www.techradar.com/news/project-xcloud-everything-we-know-about-microsofts-cloud-streaming-service.

[45] N. Mohamed, J. Al-Jaroodi, and I. Jawhar. Utilizing fog computing for multi-robot systems. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 102–105, Jan 2018.

[46] N. Mohan and J. Kangasharju. Edge-fog cloud: A distributed cloud for internet of things computations. In *2016 Cloudification of the Internet of Things (CIoT)*, pages 1–6, Nov 2016.

[47] N. Mohan, A. Zavodovski, P. Zhou, and J. Kangasharju. Anveshak: Placing edge servers in the wild. In *Proceedings of the 2018 Workshop on Mobile Edge Communications*, pages 7–12, 2018.

[48] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara. Cloudpath: A multi-tier cloud computing framework. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 20. ACM, 2017.

[49] A. Narayanan, E. Ramadan, J. Carpenter, Q. Liu, Y. Liu, F. Qian, and Z.-L. Zhang. A first look at commercial 5g performance on smartphones. In *Proceedings of The Web Conference 2020*, WWW '20, page 894–905, New York, NY, USA, 2020. Association for Computing Machinery.

[50] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, and J. Van der Merwe. Towards understanding tcp performance on lte/epc mobile networks. In *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*. ACM, 2014.

[51] S. A. Noghabi, L. Cox, S. Agarwal, and G. Ananthanarayanan. The emerging landscape of edge computing. *GetMobile: Mobile Comp. and Comm.*, 23(4):11–20, May 2020.

[52] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay, and A. Vahdat. Democratizing the network edge. *SIGCOMM*

Comput. Commun. Rev., 49(2):31–36, May 2019.

[53] J. S. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis. The benefits of self-awareness and attention in fog and mist computing. *Computer*, 48(7):37–45, 2015.

[54] K. Raaen, R. Eg, and C. Griwodz. Can gamers detect cloud delay? In *2014 13th Annual Workshop on Network and Systems Support for Games*, pages 1–3. IEEE, 2014.

[55] S. Rambo and E. Sperling. Racing To The Edge. "https://semiengineering.com/racing-to-the-edge/".

[56] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference*, IMC '19, pages 267–279, New York, NY, USA, 2019. Association for Computing Machinery.

[57] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis. In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, pages 150–156, New York, NY, USA, 2017. ACM.

[58] M. Satyanarayanan. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Computing and Communications*, 18(4):19–23, 2015.

[59] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, (4):14–23, 2009.

[60] B. Schlinker, I. Cunha, Y.-C. Chiu, S. Sundaresan, and E. Katz-Bassett. Internet performance from facebook's edge. In *Proceedings of the Internet Measurement Conference*, IMC '19, page 179–194, New York, NY, USA, 2019. Association for Computing Machinery.

[61] R. Singh, A. Dunna, and P. Gill. Characterizing the deployment and performance of multi-cdns. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, pages 168–174, New York, NY, USA, 2018. Association for Computing Machinery.

[62] R. Staff. RIPE Atlas: A global internet measurement network. *Internet Protocol Journal*, 18(3), 2015.

[63] Statista. The statistics portal for market data. "https://www.statista.com/", 2019.

[64] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai. Multi-path multi-tier 360-degree video streaming in 5g networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 162–173, New York, NY, USA, 2018. ACM.

[65] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Measuring home broadband performance. *Commun. ACM*, 55(11):100–109, Nov. 2012.

[66] S. Sundaresan, N. Feamster, and R. Teixeira. Home network or access link? locating last-mile downstream throughput bottlenecks. In *International Conference on Passive and Active Network Measurement*, pages 111–123. Springer, 2016.

[67] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[68] TeleGeography. Submarine Cable Map. "https://www.submarinecablemap.com/", 2019.

[69] The New York Times. Inside China's Dystopian Dreams: A.I., Shame and Lots of Cameras. "https://www.nytimes.com/2018/07/08/business/china-surveillance-technology.html", 2019.

[70] M. Trevisan, D. Giordano, I. Drago, M. Mellia, and M. Munafo. Five years at the edge: Watching internet from the isp network. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, pages 1–12, New York, NY, USA, 2018. ACM.

[71] Verizon. Customers in Chicago and Minneapolis are first in the world to get 5G-enabled smartphones connected to a 5G network. https://www.verizon.com/about/news/customers-chicago-and-minneapolis-are-first-world-get-5g-enabled-smartphones-connected-5g.

[72] J. Wang, Y. Zheng, Y. Ni, C. Xu, F. Qian, W. Li, W. Jiang, Y. Cheng, Z. Cheng, Y. Li, et al. An active-passive measurement study of tcp performance over lte on high-speed rails. *International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[73] D. L. Woods, J. M. Wyma, E. W. Yund, T. J. Herron, and B. Reed. Factors influencing the latency of simple reaction time. *Frontiers in human neuroscience*, 9:131, 2015.

[74] A. Zavodovski, S. Bayhan, N. Mohan, P. Zhou, W. Wong, and J. Kangasharju. Decloud: truthful decentralized double auction for edge clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 2157–2167. IEEE, 2019.

[75] A. Zavodovski, N. Mohan, S. Bayhan, W. Wong, and J. Kangasharju. ICON: Intelligent Container Overlays. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, HotNets '18, pages 15–21, New York, NY, USA, 2018. ACM.

Paper IV

Surrounded by the Clouds

A Comprehensive Cloud Reachability Study

Lorenzo Corneo
Uppsala University

Maximilian Eder
Technical University of Munich

Nitinder Mohan
Technical University of Munich

Aleksandr Zavodovski
Uppsala University

Suzan Bayhan
University of Twente

Walter Wong
University of Helsinki

Per Gunningberg
Uppsala University

Jussi Kangasharju
University of Helsinki

Jörg Ott
Technical University of Munich

ABSTRACT

In the early days of cloud computing, datacenters were sparsely deployed at distant locations far from end-users with high end-to-end communication latency. However, today's cloud datacenters have become more geographically spread, the bandwidth of the networks keeps increasing, pushing the end-users latency down. In this paper, we provide a comprehensive cloud reachability study as we perform extensive global client-to-cloud latency measurements towards 189 datacenters from all major cloud providers. We leverage the well-known measurement platform RIPE Atlas, involving up to 8500 probes deployed in heterogeneous environments, e.g., home and offices. Our goal is to evaluate the suitability of modern cloud environments for various current and predicted applications. We achieve this by comparing our latency measurements against known human perception thresholds and are able to draw inferences on the suitability of current clouds for novel applications, such as augmented reality. Our results indicate that the current cloud coverage can easily support several latency-critical applications, like cloud gaming, for the majority of the world's population.

CCS CONCEPTS

• **Networks** → **Network measurement**; *Public Internet*.

KEYWORDS

Cloud reachability; Internet measurements

ACM Reference Format:
Lorenzo Corneo, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. 2021. Surrounded by the Clouds: A Comprehensive Cloud Reachability Study. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449854

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449854

1 INTRODUCTION

During the last decade, cloud computing has gained a central role in many networked services over the Internet. According to Gartner [13], the cloud services market grossed $242.7 billion in 2019 and is expected to grow by 6.3% in 2020. This computing paradigm became so popular due to its ability to provide seemingly unlimited storage and computational capabilities through its highly efficient and optimized hardware infrastructure. In the early days of cloud computing, commodity equipment could not compare to datacenters' powerful hardware as it would have required considerable purchasing expenses. Cloud computing provided a way to reduce complex computation times dramatically. Additionally, the storage functionality allowed users to synchronize personal data over multiple devices. Cloud computing, through its appealing and flexible pricing models (e.g., pay-as-you-go [12, 15] and transient virtual machines [32]), relieves businesses, institutions, and individuals from equipment investments for storage and computation.

Since 2009, cloud computing has been challenged by the advent of edge computing, a new computing paradigm that has become very popular and well received by both industry [4] and academia [24, 30]. In fact, the research community started questioning the general applicability of cloud computing with respect to emerging enabling technologies and novel applications, such as augmented reality, industrial Internet of Things, etc. The primary motivating assumption within the edge computing community is rather long end-to-end cloud access latency due to limited and sparse deployment of datacenters across the globe. As a result, next-generation networked applications cannot meet their latency requirements while operating over the cloud infrastructure.

However, since 2009, several trends in networking and IT have drastically changed the reach of cloud computing. Cloud providers have expanded their geographical coverage by extensively establishing cloud regions in different parts of the globe while preserving their key success enabler – economies-of-scale. For example, Amazon's cloud network has expanded from 3 to 16 countries, with 22 newly built datacenters, over the last decade. Consequently, cloud providers can now support computationally complex tasks, such as voice assistance services like Siri or Cortana, without noticeable delays. Moreover, a number of recent latency-critical applications, e.g., cloud-based gaming [14, 22], backed by major cloud providers, are already available in the market. Such offerings largely rely on the throughput of the underlying networks, which continue to show steady growth.

We believe that, driven by the enthusiasm for newer computing paradigms, both practitioners and researchers of edge computing may have missed the significant efforts of cloud providers to become more and more pervasive towards the end-users. Interestingly, very little attention has been paid to quantify the reach and (consequently) applicability of current cloud infrastructure for latency-critical applications. Related works on this subject are either too dated [20] or focus on a single cloud provider [19]. In this work, we fill this chasm by expanding on our previous work [23] and present a comprehensive global cloud reachability study – aimed to estimate the cloud access latency and the path length between end-users and the datacenters. The key contributions we make in this paper are as follows:

(1) We conduct a large-scale measurement study over the span of 12 months, analyzing the reachability of ten major cloud networks – totaling 189 datacenters deployed in 28 countries. We use more than 8000 RIPE Atlas probes deployed in 184 countries to periodically measure user-to-cloud latency (`ping`) and path length (`traceroute`) over ICMP and TCP. Our collected dataset reaches almost 60 GB in size and is publicly available at [9].
(2) We analyze the spread of the clouds globally and identify their suitability for deploying latency-critical applications in the cloud. We do this by comparing the obtained latency distributions against three well-known timing thresholds, namely, human reaction time, human perceivable latency, and motion-to-photon. Throughout the paper, we compare our results with these timing thresholds to provide an application-centric perspective. Further, we take a closer look at the cloud reachability in the US and in Asia: two regions with different datacenters coverage as well as different networking infrastructures.
(3) We conduct a thorough user-to-cloud path analysis to showcase the extent of cloud pervasiveness over the Internet. Our results show that cloud providers that deploy their private wide area network (WAN) exhibit a high level of cloud pervasiveness. Conversely, cloud providers that depend on public Internet have a low level of pervasiveness. Furthermore, while we find latency differences between private and public WANs to be comparable, providers on the public WAN deliver higher latency variation, when compared to providers with their own network infrastructure.
(4) Supported by our large scale dataset, we also present a plausible road map for future cloud deployment strategies. Our findings show that cloud deployment in continents such as North America, Europe, and Oceania would bring little benefit to the end-user latency because of the existing high density of datacenter deployments. In contrast, Asia, South America, and Africa can benefit greatly from increased cloud deployment and show the largest potential in latency gains.

The remainder of this paper is organized as follows. Section 2 describes the three latency concepts we use to analyze the performance of cloud in meeting the latency requirements. Section 3 introduces our measurement methodology, while Section 4 presents our findings on cloud reachability via our measurements. It is worth noting that this work and the dataset we collected does not raise any ethical issues. In Section 5, we discuss the implications of our study as well as its limitations, while we provide the related work in Section 6. Section 7 concludes our paper.

2 BACKGROUND

In this section, we describe three well-known timing thresholds that we use to quantify the level of cloud reachability across the world. We match these thresholds to the requirements of current and future networked applications that demand strict latency requirements for operation. As a result, we study cloud reachability from the perspective of understanding if current cloud infrastructure is a feasible option for supporting near-future applications.

(1) Motion-to-Photon (MTP) is the delay between user input and its effect to be reflected on a display screen. MTP is guided by the human vestibular system, which requires sensory inputs and interactions to be in complete sync, failure of which results in motion sickness and dizziness. Maintaining latency below MTP, i.e., $\lesssim$ **20 ms**, is key for immersive applications, such as AR/VR, 360° streaming, etc. [21]. Of this, $\approx$ 13 ms can be taken up by the display technology due to refresh rate, pixel switching, etc., which leaves a budget of $\approx$ 7 ms for computing and rendering (including RTT to compute server) [7].

(2) Human Perceivable Latency (HPL) threshold is reached if the delay between user input and visual feedback becomes large enough to be detected by the human eye [27]. HPL threshold plays a key role in the QoE of applications where the user interaction with the system is fully or semi-passive, e.g., video streaming (stuttering), cloud gaming (input lags), etc. HPL is roughly estimated to be **100 ms**.

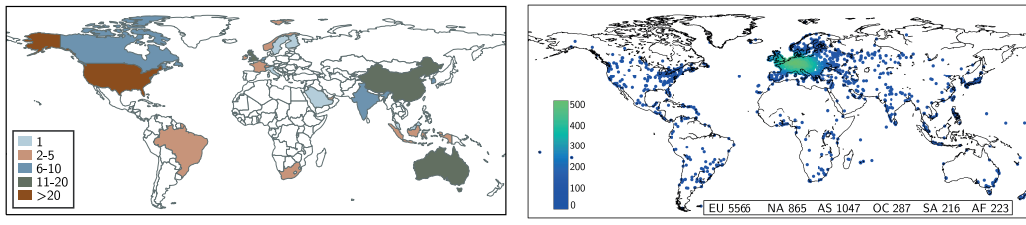
(3) Human Reaction Time (HRT) is the delay between the presentation of a stimulus and the associated motor response by a human. While HRT is highly dependent on the individual (and can be improved by training), its value is reported to be $\approx$ **250 ms** [37]. Latencies for applications that require active human engagement, such as remote surgery, teleoperated vehicles, etc., must operate within HRT bounds.

3 MEASUREMENT METHODOLOGY

In this section, we describe our methodology for measuring cloud reachability across the globe. We begin by introducing our selection criteria for targeted datacenters and vantage points, followed by a description of our experiments.

Table 1: Global density of cloud endpoints and their backbone infrastructure type used in our measurements.

	Datacenters per continent						Backbone
	EU	NA	SA	AS	AF	OC	N/W
Amazon EC2 (AMZN)	6	6	1	6	1	1	Private
Google (GCP)	6	10	1	8	-	1	Private
Microsoft (MSFT)	12	10	1	11	2	4	Private
Digital Ocean (DO)	4	6	-	1	-	-	Semi
Alibaba (BABA)	2	2	-	16	-	1	Semi
Vultr (VLTR)	4	9	-	1	-	1	Public
Linode (LIN)	2	5	-	3	-	1	Public
AMZN Lightsail (LTSL)	4	4	-	4	-	1	Private
Oracle (ORCL)	4	4	1	7	-	2	Private
IBM (IBM)	6	6	-	1	-	-	Semi
Total	**50**	**62**	**4**	**58**	**3**	**12**	



(a) Distribution of datacenters operated by nine major cloud providers (refer to Table 1 for per-provider distribution).

(b) Distribution of 8000+ RIPE Atlas probes used in our measurements.

Figure 1: Global coverage of our measurement setup. Cloud datacenters in (a) represent our endpoints, and RIPE Atlas probes in (b) are the vantage points for our measurements.

3.1 End-Points Selection

We chose datacenters from *nine* different cloud providers as endpoints, namely, Amazon, Google, Microsoft Azure, IBM, Oracle, Alibaba, Digital Ocean, Linode, and Vultr. For Amazon, we chose both its EC2 and Lightsail offerings. The chosen operators are widely used, well-established, and provide global coverage with a distinct infrastructure, that is, their backbones could be either private or public. For every cloud provider, we retrieved the host name of a public virtual machine hosted by CloudHarmony [8]. In total, our dataset includes 189 cloud region end-points as targets, the geo-distribution of which is shown in Figure 1(a). Moreover, Table 1 shows the distribution of the datacenters in our dataset by cloud provider and deployed continent.

Besides global coverage, cloud performance is also heavily influenced by the network between users and datacenters, and between datacenters. Some providers, e.g., Linode, have set up their datacenters as independent "islands" and largely rely on the *public Internet* for inter-datacenter connectivity. On the other hand, providers such as Amazon have set up *private*, large-bandwidth, low latency network backbones to interconnect all their datacenters [31]. Additionally, several cloud providers also sign agreements with major ISPs operating globally to enable direct peering between the ISP gateway and their private point-of-presence (PoP) [2]. This allows end-users to avoid the public Internet completely while transiting to cloud network. For instance, a recent study shows that Google peers with more than 5700 ASes around the globe, and the number has been increasing consistently every month [6]. Table 1 also lists whether a cloud provider has a fully-private (*Private*), private within a continent (*Semi*), or a public Internet based (*Public*) network backbone.

3.2 Vantage Points Selection

Our vantage points are probes from the RIPE Atlas platform [33], which is a *de-facto* standard for conducting measurements within the network research community. RIPE Atlas is a global Internet measurement network, especially used for reachability, connectivity, and performance studies. The platform includes thousands of small hardware probes[1] connected to the Internet all over the globe. Users can perform active network measurements (`ping` or `traceroute`, etc.) using these probes to end-points of their choice. Atlas probes are installed in heterogeneous network environments, such as core, access, or home networks, allowing us to analyze the reachability of cloud datacenters globally. Despite Atlas's dense deployment, many of the probes are hosted by cloud and network providers – allowing them to monitor their network reachability from outside their infrastructure [3]. Since these probes do not reflect the user connectivity and have the potential to add bias to our measurements, we manually filter out all such probes from our measurements using their user-defined tags [29] (e.g., `datacentre`, `us-east*`, `us-west*`, `gcp`, and `aws`, etc.). This left us with more than 8000 probes distributed in 184 countries across the globe. Figure 1(b) shows the geo-distribution of the probes used in our experiments. The majority of the selected probes are located in Europe and North America (33.5% and 26.5%), which allowed us to exhaustively analyze the performance of the bulk of datacenter deployment on the same continents.

3.3 Experiments

Our objective was to analyze *two* key aspects of cloud reachability: (i) *user-to-cloud latency* and (ii) *path lengths*. Both our experiments ran in parallel from *September 2019* to *September 2020*, resulting in an ≈60 GB dataset. Our collected data is publicly available at [9].

(i) Latency Estimation. We estimate end-to-end latencies between users and cloud datacenters via `ping` measurements. We configured the Atlas probes to `ping` all available datacenters within the same continent every 3 hours throughout the measurement period. For probes in continents with low datacenter density, e.g., Africa and South America, we also included `ping` latencies to datacenters in adjacent continents, i.e., Europe and North America, respectively. We augment the latencies from ICMP-based pings by those from TCP traceroute, see (ii).

(ii) Path Length Estimation. We estimate the end-to-end distance (as hop count) between users and cloud datacenters via `traceroute`

[1] RIPE Atlas now also integrates software probes but they were not yet available at the time this study was carried out.

measurements (§ 4.2) repeated on a daily basis. In addition to ICMP-based traceroutes, we also launched `TCP traceroute` and record per-hop latency. Unlike ICMP, our TCP measurements are guaranteed to be end-to-end and provide us with an accurate representation of connection latencies encountered by real applications operating in the cloud. The latency in the last-hop of TCP `traceroute` characterizes probe-to-VM RTT, which we use to augment our latency measurements from `ping`. Unlike our latency measurement setup, we configured Atlas probes to record `traceroutes` towards *all* datacenter endpoints. As a result, we were able to identify many unique paths from users to cloud in our resulting dataset - specifically more than 450,000 in the US, 8345 in South America, nearly 3 million in Europe, over 630,000 in Asia, and 6880 in Africa. We removed any unresponsive hops and private IP addresses in our processing phase since they depend only on the internal LAN configuration and are not part of the public Internet.

(iii) Network composition. To further quantify the footprint of cloud providers and identify several organizations that operate on a user's path to the cloud, we first map the Autonomous System Number (ASN) with IP address of every hop recorded in our `traceroute` measurement using PyASN [16]. Further, we query PeeringDB [26] dataset, which provides us with the name, location, and network type of organizations operating on the path.

Experiment configuration. In order to ensure that our analysis is statistically significant, we calculate the minimum measurement sample size required for each country. We define the required confidence interval for the measurement as $n = \frac{z^2 \times \hat{p}(1-\hat{p})}{\epsilon^2}$, where z is the *z-score*, $\hat{p}$ is the population proportion, n is the target sample size, and ϵ is the margin of error. Therefore, for an interval of confidence of 95% and an error of $\epsilon = 2\%$, we collect at least 2400 measurements per country. Furthermore, while comparing the end-to-end latencies from our ICMP and TCP measurements, we found ICMP values to be consistently larger than TCP. This was the case in Asia, Europe, Oceania and South America. On the other hand, TCP exhibited larger distribution in Africa an North America, even though the median RTT is comparable with ICMP. We believe this happens because ICMP packets are often treated as low priority by cloud organization's firewall and can be treated differently than regular application packets (like HTTP), which use TCP as the underlying protocol. Therefore, while our TCP measurements closely mimic application connectivity latencies, our ICMP-based measurements represent the worst-case connectivity between user and cloud. A more extensive comparison of the differences between TCP and ICMP is left for future study.

4 MEASUREMENTS ANALYSIS

In this section, we offer a two-fold analysis mainly addressed to estimate the pervasiveness of cloud datacenters, from the point of view of access latency and access path length.

4.1 Cloud Access Latency

The Potential. We begin by showcasing *the least possible latency for a user to access the closest datacenter across the globe.* We extract the minimum `ping` latency observed by the best-performing probe for every country to any cloud datacenter. Figure 2 shows the

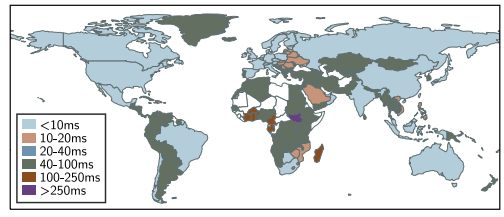


Figure 2: Minimum latency to datacenters observed across the globe.

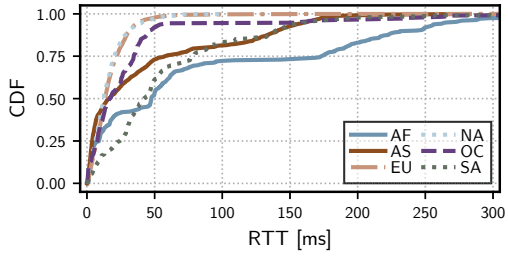


Figure 3: Distribution of minimum RTT by all probes to the nearest datacenter grouped by continent.

distribution of latency per country as a heatmap. The results show that 45 countries can access a cloud datacenter with RTTs less than 10ms, and 21 countries with RTTs between 10 to 20 ms. While our results are "optimistic" – in that they show the minimum latency – they also indicate that the cloud potentially is able to provide latency within the boundaries of MTP to the majority of the world. We will revisit this issue later in the paper.

Our findings become more intuitive when considering the density of datacenters across the globe, as shown in Figure 1(a). Countries with access latency less than 10 ms typically have a local datacenter (one or more), which offers very low latency if accessed from a managed (public or otherwise) network. Some countries, such as the US, UK, Japan, and India have more than one datacenter deployed by the same provider. Countries with access latencies less than 20 ms either share borders or have direct fiber connectivity [34] to the country housing a datacenter. Out of the rest, 49 countries have latencies between 20-40 ms and 53 between 40-100 ms. Note that probes in all *but* 16 countries (majorly in Africa) can potentially access a cloud datacenter within HPL bounds (100 ms).

Figure 3 depicts the smallest latency distribution experienced by every probe to any datacenter, grouped by continents. Note that while the measurements are largely from probes deployed in home networks, they also include probes which may not have a stable Internet connection. Despite this, the results look quite in favor of the cloud and support the findings in Figure 2. Around 80% of probes in Europe and North America, which is around 50% of the total

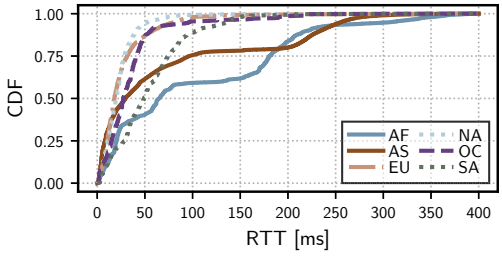


Figure 4: Distribution of all RTT values recorded from all Atlas probes in our dataset to the closest datacenter.

number of probes used in our experiments, can access a datacenter within 20 ms. Probes in Oceania follow a similar performance pattern as almost all of them can access the cloud within 50 ms RTT. Surprisingly, despite the low density of available datacenters and substandard network deployment, 75% of probes in Africa and Latin America achieve less than 100 ms access latency to the cloud, thereby meeting HPL requirements. Almost all, but a few probes in Africa, can reach the cloud within HRT threshold (i.e., 250 ms).

Takeaway – 168 countries out of a total of 184 in our dataset can support applications bounded by human perception. All probes (excluding long tails) in North America, South America, Europe, and Oceania can reach the cloud within 90 ms. Moreover, slightly more than 75% of the probes in Asia and Africa satisfy the HPL threshold.

The Reality. Till now, our latency analysis focused on the best-case scenarios to illustrate the potential reach of the cloud. We now turn to our entire latency dataset to showcase what the reality of cloud reachability is today. Figure 4 shows a comprehensive view of our latency dataset. We show the distribution of *all* latencies observed by probes to their nearest datacenter throughout our measurements duration of 12 months.

It is evident that probes in North America, Europe, and Oceania exhibit excellent cloud reachability. *More than 75% of the total probes* in all three continents have RTT to the cloud within the HPL. A closer look reveals that the top 25% of connected probes in North America and Europe can reach the cloud within the bounds of MTP latency, indicating that the cloud can support emerging applications such as AR/VR and autonomous vehicles. The reason for this exceptional performance, as made evident from the previous section, is the concentrated efforts of cloud providers to deploy datacenters throughout the US and central Europe. Additionally, thanks to the vast number of ISPs operating in these two continents, the majority of users can consistently connect to the cloud via high-bandwidth, low-latency fiber connections. However, note the long tail of latency distribution for Europe, which is largely missing from North America. The cause is the absence of a local datacenter or high latency to connect to the one located in a neighbouring country. The result is in line with our initial assessment of Figure 2, where the bulk of countries exhibiting high access latencies did not have a datacenter in close proximity.

We now focus on the remaining continents, i.e., South America, Asia, and Africa. Cloud reachability from within these continents is quite poor as only a fraction of probes are able to satisfy the 100 ms HPL threshold. Probes in Asia show very diverse latencies primarily due to scattered datacenter deployment favoring certain countries, like China and India. Unsurprisingly, the worst performance is in Africa as the continent is severely under-served, both in terms of cloud presence (only three operating datacenter in South Africa) and reliable network infrastructure [5].

Takeaway – North America, Europe, and Oceania easily satisfy the HPL, and almost 25% even support MTP latency. On the other hand, Asia, South America, and Africa show considerably longer latencies to the cloud due to a lack of extensive cloud and network infrastructure deployment.

Wide Area Network Latency Differences. We now assess the impact of network backbone infrastructure on cloud reachability performance. As summarized in § 3.1, many cloud providers deploy extensive private wide area network (WAN) to interconnect their datacenters that provide clients fast-track paths to services hosted in their infrastructure. Table 1 enlists the network backbone type used by different cloud providers targeted in our measurements. Figure 5 shows the distribution of the latencies achieved by Atlas probes, at continental granularity, towards the closest datacenter of every cloud provider. Since the aim of this work is not to provide a benchmark study comparing the performance of different cloud operators across the globe, we do not probe all cloud regions in this analysis. Instead, we only draw results from those regions which were found closest (in latency) to our vantage points.

From the figure, it is evident that the availability of private network backbone in continents with extensive network deployment, like North America and Europe, does not seem to have much impact on cloud reachability. In fact, we find that all cloud providers exhibit similar latency distributions in these regions – accentuated more towards providers relying on the public Internet. In Oceania, Amazon EC2, Alibaba Cloud, and Oracle achieve the least latency results while Microsoft Azure and Linode perform similarly but with higher variance. We justify their superior performance to their extensive deployment in the continent. Within Asia, almost all providers perform similarly, and we do not observe any significant benefits favoring providers with private WAN and those relying on public Internet. For South America, we probed Amazon EC2, Microsoft, and Oracle since only those have datacenters deployed within the continent. For the rest of the providers, we show latencies from South American probes to their datacenters in North America. We observe that cloud providers with local datacenter deployment perform significantly better than those with infrastructure in the neighbouring continent. A similar trend can also be observed in Africa, where providers with in-land datacenter deployment (specifically Microsoft and Amazon EC2) show much lower latency than their counterparts, which host datacenters in the neighbouring continent of Europe. It is to be noted that we draw our inferences from small-sized ICMP and TCP packets, and the impact of private backbone will be far more significant for elephant flows within the cloud infrastructure.

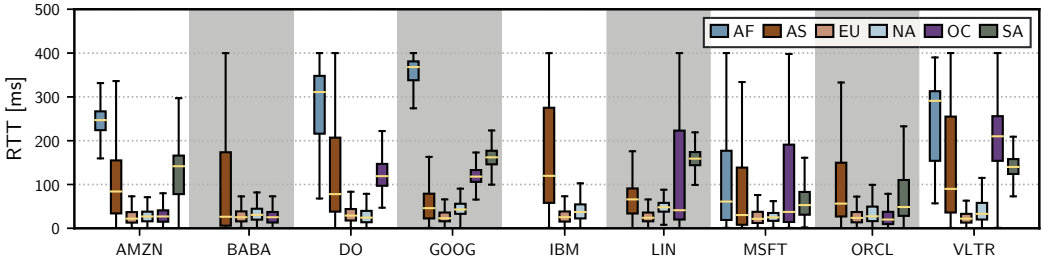
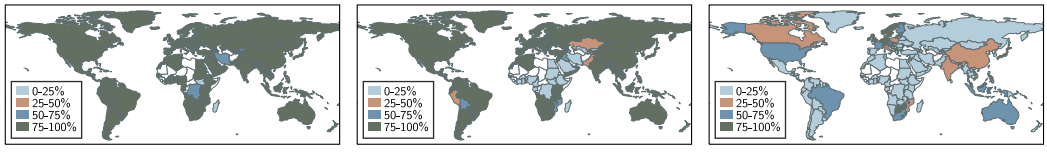


Figure 5: Cloud access latency to the closest cloud datacenter of every provider in different continents.



(a) Probes RTT within HRT. **(b) Probes RTT within HPL.** **(c) Probes RTT within MTP.**

Figure 6: Global coverage of our measurement with respect to the three timing thresholds defined in §2

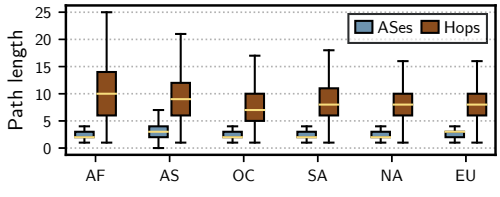


Figure 7: Path length to the closest cloud.

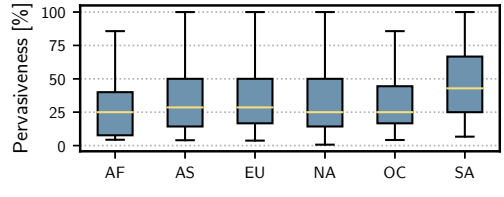


Figure 8: Per-continent cloud pervasiveness.

Takeaway – The impact of private WAN availability on cloud reachability is not as significant as otherwise assumed. In continents with dense network deployment, public Internet delivers almost similar performance compared to a cloud provider that deploys its own private network backbones. Moreover, the availability of datacenters within a continent impacts connectivity far more than the type of interconnecting network infrastructure.

Cloud Application Readiness. We conclude this section by investigating *cloud maturity level* – the state of global cloud connectivity (at country-level granularity) to achieve the timing thresholds discussed in § 2, i.e., MTP, HPL, and HRT. Figure 6 illustrates the global RTT distribution from all probes in our dataset, one for each timing threshold. Different color groups denote different percentiles of the distribution. The results suggest that almost every country across the globe can consistently reach the closest cloud datacenter within the boundaries of the HRT. In fact, only two (out of 184) countries in our dataset achieve HRT less than 25% of the times, and three countries lie between 50 and 75%. For HPL, we observe that the cloud maturity level changes slightly compared to the HRT. The distribution of RTTs degrades only in certain countries, mainly clustered in Central Africa, the Middle East, and South America. Specifically, 140 countries achieve RTTs consistently within the boundaries of the HPL, six achieve that only 50 to 75% of the times, another six within 25 to 50% and, 16 countries fail to reliably meet the HPL threshold. The distribution changes substantially for the MTP threshold, where only 24 countries can consistently meet the timing deadline (75–100%). Conversely, 125 countries outrightly fail to meet the threshold (0–25%), while the remaining 25 countries can reach cloud within MTP between 25–75% of the times.

Takeaway – The current cloud infrastructure is able to deliver network latency compliant with both HRT and HPL safely. However, only a small minority of countries reliably meet the MTP threshold suggesting that either cloud deployment or network should be improved.

4.2 Cloud Access Path Length

Distance to the Cloud. We complement our latency analysis in the previous section by investigating path lengths to the cloud.

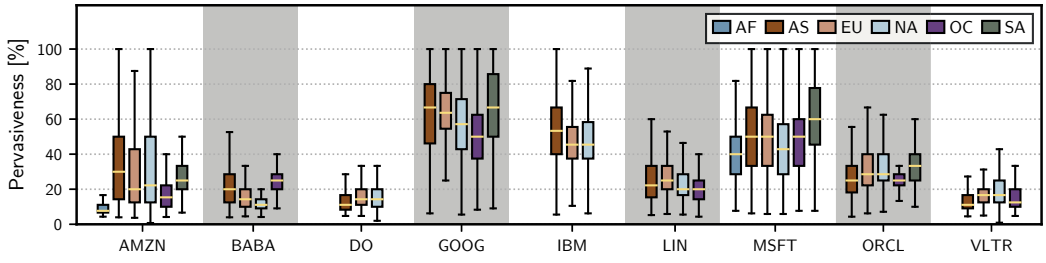


Figure 9: Degree of the pervasiveness of the cloud providers in different continents.

The study allows us to better understand the state of user-to-cloud connectivity from different parts of the globe over the Internet. We exploit our `traceroute` measurements (see § 3.3 (ii)) and extract distance from probe to the closest cloud datacenter (in terms of routers), and organizations operating those routers (in terms of ASNs). We derive the latter by mapping the IP addresses of on-path routers to their ASN numbers and further correlating them with distinct organizations using PeeringDB [26]. Figure 7 shows our results, specifically the number of routers and ASNs on a path between a probe and its nearest datacenter in every continent. Our key findings are as follows. End-user paths to the cloud can range anywhere between 7–10 hops on average and are shorter in continents with extensive cloud presence (e.g., NA and EU). However, most of these routers belong to a very small set of ASNs; usually, the resulting paths connecting these routers are managed by large network operators as well as cloud providers, and are highly optimized. The largest chasm between the number of routers and ASNs exists in Africa, showing the presence of long but managed paths to the cloud (some even traversing long transatlantic links to connect to datacenters located in NA). Overall, across the globe, a typical user can traverse 3–4 ASNs, on average, before reaching the nearest cloud region.

Takeaway – End-user path to the cloud is still quite long (in number of hops). However, these long paths are operated by a few organizations showcasing a highly managed cloud network connectivity.

Pervasiveness of the Cloud. As we are interested in understanding the degree of the pervasiveness of cloud networks, we hereby investigate *how much of the user-to-cloud path is owned by cloud providers*. We define *cloud pervasiveness* as the ratio between the number of routers owned by the cloud providers and the overall path length to the cloud. High pervasiveness indicates that the cloud providers are very close to the end-users and, conversely, a low ratio translates into cloud providers being faraway.

Figure 8 shows the extent of cloud pervasiveness of the closest datacenter for all continents. We can observe that the cloud providers already own 20–40% of the user-to-cloud path on average. It is also quite common for the cloud to own and operate upwards of 50% of the path, often reaching 100% in some regions. This indicates that the first public IP address encountered by the probe is entry to the cloud network. We found this phenomenon to be a common occurrence for probes installed in cities with a colocated datacenter. On the other hand, some cloud providers rely on the public Internet and do not own a private WAN (see Table 1). Consequently, these providers only own the final hop, thus pushing the distribution towards the lower end. To understand this further, we investigate the impact of private and public WANs on cloud provider's pervasiveness.

Figure 9 depicts the continental cloud pervasiveness grouped by providers. The figure provides immediate visual feedback for distinguishing providers with private WANs from those relying on the public Internet. Providers using the latter have a level of pervasiveness – constantly below and capped at 50%. On the other hand, Amazon, Google, IBM, and Microsoft exhibit a high degree of cloud pervasiveness by abundantly owning a majority of routers in the paths user-to-cloud. Note that the distributions of Amazon through Africa, Oceania, and South America are skewed as those measurements also target its datacenters in neighboring continents.

Takeaway – Cloud connectivity has become highly pervasive across the globe, with providers installing managed network infrastructures and establishing peering agreements with ISPs in the region. Of these, providers that make use of privately owned networks exhibit a high degree of pervasiveness. Conversely, providers relying on the public Internet have a low level of pervasiveness.

4.3 Cloud Access Case Studies

Case Study A: The United States of America. We now investigate the extent of cloud reachability by users within the United States of America. We find the US as a good object of study since it covers a large geographical area, has a large population, and has remained the focal point for major cloud providers – as reflected by the dense cloud presence within the country (Table 1). We selected the most populated regions in the US using the US Primary Statistical Areas (PSA) [10, 36]. The federal government has defined 100 PSAs, which collectively house more than 80% of the total US population. We further selected 93 PSAs (7 PSAs did not have any functioning RIPE Atlas probe) and collected up to 25 probes within a radius of 125 km from the center of PSA location. Overall, we selected 701 probes, each performing multiple `ping` measurements towards 15 datacenters belonging to all cloud providers within the US. Figure 10 shows the results.

We show the minimum, median, and 95th percentile of latency observed in every PSA. The distribution is weighted according to

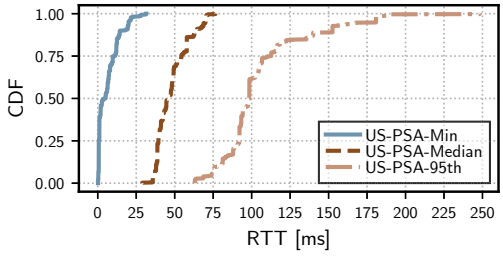


Figure 10: Distribution of RTT in US primary statistical areas weighted by normalized population.

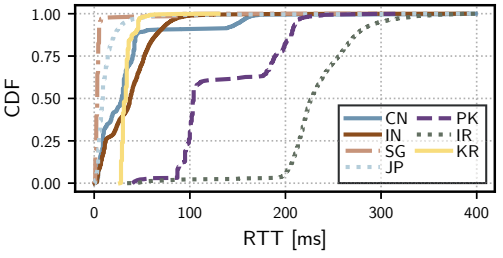


Figure 11: Distribution of RTT in some Asian countries with and without in-country datacenters.

the population of each PSA; visually, this translates to a higher vertical step in the CDF, for a larger population. The median distribution shows that almost the entire US population has median access latency below 75 ms – well within the HPL threshold. The differences, however, show up for the 95*th* percentile distribution of PSAs latency as it includes probes installed in imperfect network conditions. Even in this case, ≈ 60% of the US population can reach the cloud within the coveted 100 ms threshold.

Case Study B: Asia. We contrast our analysis above by focusing on the state of cloud access in Asia. As previously hinted in § 4.1, latency distributions in Asia are rather skewed, resulting in unequal performance throughout the continent. To further investigate the cause, we carefully select seven Asian countries based on their landmass and physical distance to the cloud. Specifically, we investigate five countries with local datacenter deployments, i.e., China (48 probes), India (108 probes), Singapore (80 probes), Korea (20 probes), and Japan (188 probes); Pakistan (12 probes), which directly shares borders with the country with datacenter (India), and Iran (120 probes), which is farthest from any datacenter in the continent, nearest deployment in UAE and India. Figure 11 shows the results.

It is evident from the figure that countries with locally deployed datacenter can consistently meet the HPL threshold (100 ms). On the other hand, the impact of large geographical distance from the nearest datacenter, becomes evident in countries with no in-land datacenter. For instance, only 40% of samples from Pakistan are below 100 ms, while the rest can only satisfy the HRT threshold (250 ms). Finally, being Iran the geographically farthest from any datacenter, the minimum latency to reach the cloud is ≈200 ms, and almost 30% of samples did not even satisfy the HRT threshold.

Takeaway – The current cloud presence in the US can easily support the bulk of emerging applications, bounded by HPL constraints, for the majority of the population. On the other hand, while cloud reachability in Asia is generally good for countries with local deployment (e.g., China, Korea, Japan, India), it gets significantly worse with increasing geographical distance from the physical location of datacenters. Furthermore, the state of the user's network connectivity does not seem to have much effect on cloud reachability, as evident from consistently high latencies achieved by probes in Iran.

5 DISCUSSION

Vantage Points Representativeness. It is well known that a vast majority of cloud-hosted applications (e.g., HTTP-based) use TCP as their transport protocol and TCP traffic dominates over other protocols over the Internet [35]. Hence, we believe that our TCP-based measurements closely reflect realistic connection establishment overheads an application would experience while connecting to the cloud. Furthermore, our 12 month data collection absorbs the impact of temporally insignificant changes on the network. On the other hand, our measurements might fall short of accounting for the factors that affect the end-to-end latency of a service, e.g., interactions within the protocol stack or amongst entities on the service delivery route, etc. The most prominent lack in our analysis is the inability to showcase the impact of queuing delays observed by regular application traffic due to significantly smaller footprint of ping packets. Hence, the latencies in this work can be viewed as the minimum end-to-end delay bound an application can observe while connecting to the cloud.

Another limitation of our study stems from our choice of measurement platform. While RIPE Atlas is considered to be a gold-standard within the Internet research community, it is also influenced by several deployment biases that readers should be aware of. Atlas probes are mostly hosted by network enthusiasts and network service providers, which can skew the availability and deployment configurations of the probe. As discussed in § 3.2, while a vast majority of Atlas probes are available in Europe and North America, only a fraction (< 10%) of the probes are deployed in Africa. Furthermore, even within Europe and North America (see Fig.1(b)), the probes are not uniformly distributed over the continent's geography. As a result, our dataset includes both countries with extremely dense probe availability and those without many options. While we compensate for some of these biases in our post-collection analysis, e.g., by carefully pruning out probes installed in privileged networks, we do not have much control over others.

Measurements Duration. *Would it be possible to obtain similar results with only one month worth of measurements?* We want to stress the fact that this question can be answered only by analyzing a long-term dataset. To provide an answer to the question, we isolated our measurements from four cloud providers: two with private WAN and two relying on the public Internet. We compared

the first four months of measurements, divided into intervals of one month. We found out that one provider with private WAN and one without exhibited very consistent and reliable connectivity. The other two providers also had very similar performance across three months (negligible change) of measurements. However, during one month (October 2019) these providers experienced significant divergence as their latency performances were rather degraded with respect to the usual. However, we have no means of ascertaining the root cause behind this. Therefore, we conclude that, most likely, we could reach similar conclusions with only 1 month of data, but this strongly depends on the time and interval of the measurements. In other words, without a known baseline, we would not know if a particular time period is "usual" or "unusual".

Where to Proceed from Here. *Can existing cloud infrastructure support the operation of future-forward latency-critical applications?* Our large-scale latency measurements have answered this question affirmatively for the majority of the continents and for the applications requiring a latency bounded by HRT. On the other hand, applications requiring latency below MTP, such as augmented reality, can only be supported, with the current cloud infrastructure, within North America, Europe and Oceania. We also found that private WANs have little-to-no impact in bringing cloud coverage any closer to users. Providers that rely on public Internet achieve almost similar latencies to those with private network backbone in regions without local datacenter deployment. However, while establishing new cloud regions globally may seem like the only viable option to drive down cloud access latencies, we also show that in regions with already high degree of cloud pervasiveness and excellent network connectivity, deploying more datacenters does not bring much benefit (e.g., the USA). Therefore, a key takeaway that existing cloud providers can take from this study could be to prioritize infrastructure expansion in under-provisioned regions, specifically Africa, Asia, and South America.

6 RELATED WORK

To the best of our knowledge, the first significant cloud reachability study dates back to 2010 [20]. However, the substantial evolution of cloud computing and datacenter deployments over the decade since its publication suggests the findings of that paper are worthwhile updating to reflect today's state of the art. More recently, the cloud performance report 2019 from ThousandEyes[2] monitors and compares 95 end-points' performance to the major cloud providers (Amazon, Microsoft, Google, Alibaba, and IBM) for a maximum period of one month during the year 2019. Their measurement methodology consists of collecting network latency and paths from 98 TCP-based vantage points in various countries worldwide. In our study, we target almost the double of endpoints (189), and we use more vantage points (up to 8500) located in 184 different countries. Furthermore, our measurements have been collected for a significantly longer period of time. For these reasons, we believe our study to be more comprehensive and broad. Furthermore, we expand our previous work [23] with more measurements and vantage points, as well as a thorough path characterization study. In particular, our cloud pervasiveness study reports similar findings to [1], where Todd et al. show that cloud providers are already bypassing Tier-1 providers, therefore making the Internet "flatter" (less hierarchical).

Related methodology partly used in our evaluation can be found in [25], [18], and [17]. In [25], the authors measured the performance of the 5G deployment in the USA. In their measurements, they selected three cloud providers (Amazon, Google, and Azure) and evaluated the base RTT without cross-traffic, and download and upload bandwidth and latency times. The results show that the 5G RTT latency has little improvement compared to 4G, and the first-hop accounts for $\sim$ 27 ms while the remaining latency in the path towards the cloud is similar to our measurements. Therefore, at this stage of 5G deployment, there is still little improvement in the last mile connectivity times, but it is expected to be addressed in the future and achieve the promised sub-millisecond RTT. In [18], the authors compare the performance of ICMP-based ping and traceroute tools to detect cloud service providers' outages. The main issue regarding ICMP measurements is that it can underestimate the availability as it only checks the network-level connectivity and not the application itself. To validate this hypothesis, the authors compare ICMP-traceroute against TCP-traceroute, and the experimental results show that there is disagreement on some measurements (up to 3%) where the ICMP fails while HTTP succeeds, and vice-versa. In [17], where the authors analyzed the inter-continental paths connecting three big cloud providers, namely Amazon, Google, and Azure. They found out that those cloud providers have dedicated paths connecting their data-centers, which increased the network path performance (lower packet loss and latency) compared to regular inter-continental data traversal through different independent ASes. We complement and confirm their analysis by adding the latency information from several Atlas probes to those cloud providers, providing a bigger picture and how close to each cloud provider each country is.

Within the same topic, in [11], the authors study cloud provider outages and dig into the causes of such events by analyzing the connectivity between major cloud service providers, e.g., Google, Amazon, Microsoft, etc. To facilitate the analysis, the authors define a set of metrics based on graph properties and measure the interconnectivity between the ASes of those cloud service providers. In [38], the authors propose a mechanism to verify whether cloud providers are respecting the subscribed SLAs for packets being processed in cloud middleboxes. In [28], the authors performed a large scale study of web page performance, showing the impact of different Web protocols and access media in the performance of page loading and overall user experience.

7 CONCLUSION

We conducted a large-scale cloud reachability study with the aim to evaluate the current state of cloud connectivity globally. In our study, we targeted 189 compute-capable cloud regions of ten major cloud networks from 8500 globally distributed RIPE Atlas probes for a period of 12 months. Through our extensive analysis of network latency, we found that the majority of the world population can access a cloud facility within 100ms – which is a critical threshold for many future-forward networked applications. Furthermore, our analysis of user-to-cloud path lengths revealed that cloud providers relying on private WAN for network interconnections are already

[2] https://www.thousandeyes.com/research/cloud-performance.

very pervasive since the majority of the paths transit through their infrastructure. However, we also found that end-to-end network latency is rarely impacted by underlying network infrastructure as even providers relying on public Internet achieve similar latencies, albeit with higher variability. Our case study analysis showcased the impact of geographical distance to cloud by analysing regions with contrasting datatcenter deployment density – the USA and Asia. Our results revealed that extensive datacenter deployment is key to make cloud access latencies consistently compatible with requirements of next-generation applications, especially for Asia, South America, and Africa.

ACKNOWLEDGMENTS

We would like to acknowledge RIPE Atlas team for providing us access to their platform and supporting our measurements with increased quota limits. This work was supported by the Swedish Foundation for Strategic Research with grant number GMT-14-0032 (Future Factories in the Cloud), the Academy of Finland in the BCDC (314167), AIDA (317086), WMD (313477) projects and Celtic project Piccolo (C2019/2-2).

REFERENCES

[1] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. 2020. Cloud Provider Connectivity in the Flat Internet. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. Association for Computing Machinery, New York, NY, USA, 230–246. https://doi.org/10.1145/3419394.3423613

[2] AWS. 2019. AWS Direct Connect Partners. "https://aws.amazon.com/directconnect/partners/". (2019).

[3] Vaibhav Bajpai, Steffie Jacob Eravuchira, and Jürgen Schönwälder. 2015. Lessons learned from using the ripe atlas platform for measurement research. *ACM SIGCOMM Computer Communication Review* 45, 3 (2015), 35–42.

[4] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*. Springer, 169–186.

[5] Marshini Chetty, Srikanth Sundaresan, Sachit Muckaden, Nick Feamster, and Enrico Calandro. 2013. Measuring Broadband Performance in South Africa. In *Proceedings of the 4th Annual Symposium on Computing for Development (ACM DEV-4 '13)*. Association for Computing Machinery, New York, NY, USA, Article Article 1, 10 pages. https://doi.org/10.1145/2537052.2537053

[6] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. 2015. Are We One Hop Away from a Better Internet?. In *Proceedings of the 2015 Internet Measurement Conference (IMC '15)*. Association for Computing Machinery, New York, NY, USA, 523–529. https://doi.org/10.1145/2815675.2815719

[7] Song-Woo Choi, Siyeong Lee, Min-Woo Seo, and Suk-Ju Kang. 2018. Time sequential motion-to-photon latency measurement system for virtual reality head-mounted displays. *Electronics* 7, 9 (2018), 171.

[8] CloudHarmony. 2020. Transparency for the cloud. "https://cloudharmony.com/". (2020).

[9] Maximilian Eder, Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, Per Gunningberg, Jussi Kangasharju, and Jörg Ott. 2021. Surrounded by the Clouds. (2021). https://doi.org/10.14459/2020mp1593899

[10] Executive Office of the President, Washington, D.C. 20503. 2013. OMB BULLETIN NO. 13-01. https://obamawhitehouse.archives.gov/sites/default/files/omb/bulletins/2013/b13-01.pdf. (28 Feb. 2013).

[11] Benjamin Fabian, Annika Baumann, and Jessika Lackner. 2015. Topological analysis of cloud service connectivity. *Computers & Industrial Engineering* 88 (2015), 151–165. https://doi.org/10.1016/j.cie.2015.06.009

[12] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. 2009. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS* 28, 13 (2009), 2009.

[13] Gartner. 2020. Gartner Forecasts Worldwide Public Cloud Revenue to Grow 6.3% in 2020. https://www.gartner.com/en/newsroom/press-releases/2020-07-23-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-6point3-percent-in-2020. (2020).

[14] Google. 2019. Stadia. https://stadia.dev/. (2019).

[15] Robert L Grossman. 2009. The case for cloud computing. *IT professional* 11, 2 (2009), 23–27.

[16] Hadi Asghari and Arman Noroozian. 2020. PyASN. "https://pypi.org/project/pyasn/". (2020).

[17] Osama Haq, Mamoon Raja, and Fahad R. Dogar. 2017. Measuring and Improving the Reliability of Wide-Area Cloud Paths. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 253–262. https://doi.org/10.1145/3038912.3052560

[18] Zi Hu, Liang Zhu, Calvin Ardi, Ethan Katz-Bassett, Harsha V. Madhyastha, John Heidemann, and Minlan Yu. 2014. The Need for End-to-End Evaluation of Cloud Availability. In *Passive and Active Measurement*, Michalis Faloutsos and Aleksandar Kuzmanovic (Eds.). Springer International Publishing, Cham, 119–130.

[19] Yuchen Jin, Sundararajan Renganathan, Ganesh Ananthanarayanan, Junchen Jiang, Venkata N. Padmanabhan, Manuel Schroder, Matt Calder, and Arvind Krishnamurthy. 2019. Zooming in on Wide-Area Latencies to a Global Cloud Provider. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 104–116. https://doi.org/10.1145/3341302.3342073

[20] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. 2010. CloudCmp: Comparing Public Cloud Providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/1879141.1879143

[21] Katerina Mania, Bernard D Adelstein, Stephen R Ellis, and Michael I Hill. 2004. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*. ACM, 39–47.

[22] Microsoft. 2019. Xbox Project xCloud. (2019). https://www.techradar.com/news/project-xcloud-everything-we-know-about-microsofts-cloud-streaming-service.

[23] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks (HotNets '20)*. Association for Computing Machinery, New York, NY, USA, 182–189. https://doi.org/10.1145/3422604.3425943

[24] Nitinder Mohan and Jussi Kangasharju. 2016. Edge-Fog cloud: A distributed cloud for Internet of Things computations. In *2016 Cloudification of the Internet of Things (CIoT)*. 1–6. https://doi.org/10.1109/CIOT.2016.7872914

[25] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. 2020. A First Look at Commercial 5G Performance on Smartphones. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 894–905. https://doi.org/10.1145/3366423.3380169

[26] PeeringDB. 2020. The Interconnection Database. https://www.peeringdb.com/. (2020). accessed on October 16, 2020.

[27] Kjetil Raaen, Ragnhild Eg, and Carsten Griwodz. 2014. Can gamers detect cloud delay?. In *2014 13th Annual Workshop on Network and Systems Support for Games*. IEEE, 1–3.

[28] Mohammad Rajiullah, Andra Lutu, Ali Safari Khatouni, Mah-Rukh Fida, Marco Mellia, Anna Brunstrom, Ozgu Alay, Stefan Alfredsson, and Vincenzo Mancuso. 2019. Web Experience in Mobile Networks: Lessons from Two Million Page Visits. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1532–1543. https://doi.org/10.1145/3308558.3313606

[29] RIPE NCC. 2020. Probe tags. "https://atlas.ripe.net/docs/probe-tags/". (2020).

[30] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. 2009. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing* 4 (2009), 14–23.

[31] Amazon Web Services. 2019. AWS Global Infrastructure Map. "https://aws.amazon.com/about-aws/global-infrastructure/". (2019).

[32] Rahul Singh, Prateek Sharma, David Irwin, Prashant Shenoy, and KK Ramakrishnan. 2014. Here today, gone tomorrow: Exploiting transient servers in datacenters. *IEEE Internet Computing* 18, 4 (2014), 22–29.

[33] RN Staff. 2015. RIPE Atlas: A global internet measurement network. *Internet Protocol Journal* 18, 3 (2015).

[34] TeleGeography. 2019. Submarine Cable Map. "https://www.submarinecablemap.com/". (2019).

[35] M. Trevisan, D. Giordano, I. Drago, M. M. Munafò, and M. Mellia. 2020. Five Years at the Edge: Watching Internet From the ISP Network. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 561–574.

[36] Wikipedia. 2019. Statistical area (United States). https://en.wikipedia.org/wiki/Statistical_area_united_states). (2019).

[37] David L Woods, John M Wyma, E William Yund, Timothy J Herron, and Bruce Reed. 2015. Factors influencing the latency of simple reaction time. *Frontiers in human neuroscience* 9 (2015), 131.

[38] X. Zhang, H. Duan, C. Wang, Q. Li, and J. Wu. 2019. Towards Verifiable Performance Measurement over In-the-Cloud Middleboxes. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1162–1170.

Paper V

(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

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

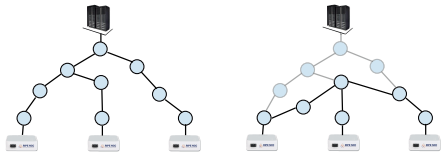
Vantage Points. We select vantage points for our measurements from RIPE Atlas [10], a *de-facto* standard, and well-established platform in the Internet measurements community.

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 `traceroutes`. 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

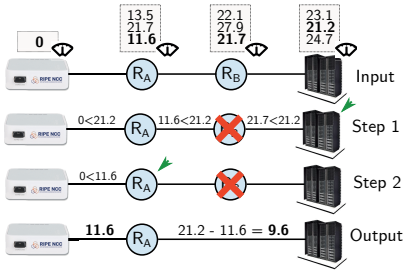


Fig. 2: Sanitizing RIPE Atlas `traceroutes` by removing nodes that violate monotonically increasing RTTs throughout the network path.

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.

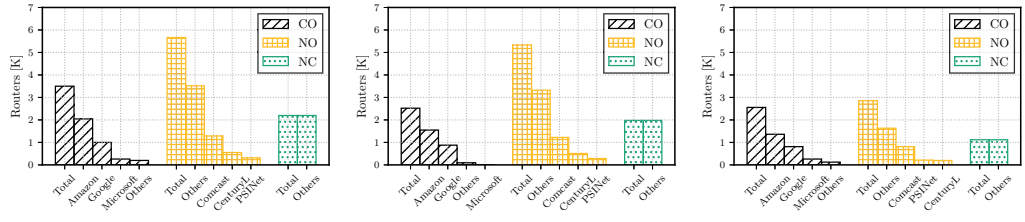
A. Owners Composition of the Network Graph

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 ≈ 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 ≈ 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.

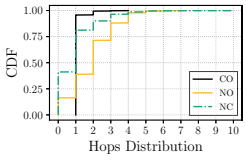
Fig. 4a shows the CDF of the number of hops (routers) belonging to each of the categories on the way to the shortest path to the closest DC. That is, if a path consists of 3 hops belonging to two NOs and one CO, three points will be mapped on the CDF, respectively NO=2, CO=1, and NC=0. From the figure, 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.

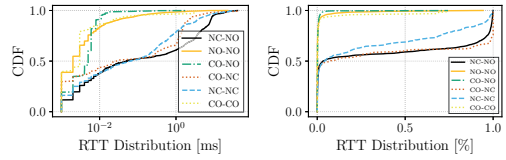
C. Latency Distribution on the Shortest Path to the Cloud

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)

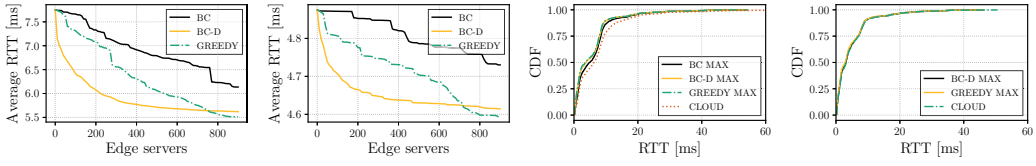
The goal of the BC strategy is to lower the end-users latency to the closest server without deploying as many servers as in the greedy strategy. *Betweenness centrality* (BC) indicates node centrality in a graph, and it is based on how many times a particular node is encountered through the shortest 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

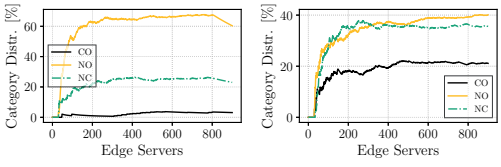
We start our analysis by calculating the access RTT for every probe. Access RTT is calculated on the shortest path toward the closest computational facility, e.g., DC or a newly placed edge server. If a probe has no edge server on the path, the access RTT equals the latency to access the nearest cloud.

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 $7.8\,\text{ms}$. On the other hand, when the number of edge servers is equivalent to the number of probes, the GREEDY strategy delivers $\approx 5.5\,\text{ms}$ while the BC-D strategy $6.7\,\text{ms}$. To put it differently, the GREEDY and BC-D strategies at peak edge deployment improves network latency by ≈40% and ≈33% respectively. More interestingly, we see that the two strategies perform equivalently when ≈ 750 servers are deployed (≈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 $4.87\,\text{ms}$. At peak edge deployment, the GREEDY strategy delivers an average access RTT of $\approx 4.6\,\text{ms}$ while the BC-D strategy $\approx 4.65\,\text{ms}$, e.g., gain of ≈6% and ≈5%, respectively. Again, the two strategies' curves intersect, meaning that they are equivalent when the number of edge servers deployed is ≈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 ≈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*.

Paper VI

Analyzing Public Cloud Connectivity at its Best

Lorenzo Corneo*
Uppsala University
Sweden

Aleksandr Zavodovski*
Uppsala University
Sweden

Andreas Johnsson
Uppsala University
Ericsson Research
Sweden

Christian Rohner
Uppsala University
Sweden

Per Gunningberg
Uppsala University
Sweden

ABSTRACT

Cloud computing has a remarkable growth not only in terms of its capacity and performance but also in its geographical coverage and density, coming closer to the users. With shorter network distances to users, it is gradually overcoming its initial drawback – the high latency of access. Thus, we are interested in quantifying the network RTT in well-connected and largely populated metropolitan areas from regular user points on the Internet. Besides RTT, we also measure its variation since it is also of paramount importance to many time-critical applications, e.g., augmented reality. In addition, we identify bottleneck links in the network paths that are responsible for RTT anomalies (e.g., by packet queuing) and hence poor performance. Our study covers over 40 metropolitan areas around the world and more than 100 centers offering public clouds. The Atlas RIPE platform is used for the measurements. The collected measurements exhibit median RTT to the nearest cloud of 6.49 ms with a standard deviation, for 50% of the samples, below 1.23 ms. We also find that cloud ingress is often responsible for the deterioration of performance.

1 INTRODUCTION

Cloud computing [1] continues to attract new businesses and customers, extending the quality and diversity of its service offerings [2, 3]. In the last decade, there has been a remarkable growth in the number of public cloud datacenters, both in increased geographical coverage as well as increased density in highly populated areas, where the number of customers can be maximized, which is far from finished. For example, Amazon has already planned an expansion of 15 new centers zone in 5 new regions [4], Microsoft Azure is planning to expand to 15 new regions [5]. Carried by such steady growth, cloud services became physically located closer and closer to the end-users. At the same time, network connectivity and performance have also improved [6]. This dilutes the cloud computing initial birth defect, namely, the high network access latency [7, 8].

Cloud datacenters have become so pervasive that the research community has started to question the role of latency reduction as a driver of edge computing [9]. This work intends to partly answer this question by systematically measure latencies to publicly available clouds in selected metropolitan areas from a large set of widespread vantage points.

The measurements are focused on areas that have a population size bigger than 1 million, and have at least one datacenter in close proximity. According to United Nations Population Division [10], over half of the global population now lives in cities. Especially high is the percentage of the urban population in Northern America (83.6%), Southern America (81.2%), and Europe (74.9%).

In fact, it is nowadays common to experience round-trip times (RTT) in the range of 20 ms to 40 ms in almost half of the world's metropolitan areas, and even below 10 ms in the US, Europe, and Asia with an abundant cloud provisioning [11].

In this work, we analyze RTT, RTT variation and identify poorly performing links that add a non-proportional large amount of variation. We conduct network measurements spanning at least 24 hours for each of the 40+ metropolitan areas under investigation. The targets of the measurements are public cloud services belonging to more than 100 datacenters, deployed in 27 countries around the world. In total, we collected more than 1.8 million measurement data points. The dataset will be publicly available. The RIPE Atlas platform [12] is used for all measurements.

The collected latency results are encouraging – the metropolitan areas under investigation deliver on the average very low latency. The median RTT for our entire dataset is 6.49 ms with a standard deviation, for 50% of the paths, of 1.23 ms. The Median Absolute Deviation (MAD) for the entire dataset is 6.85 ms (§ 3). The majority of the probes exhibit stable latency during the observed measurement period. This means that cloud datacenters in metropolitan areas are able to deliver RTTs below the motion-to-photon [13] ($\leq$ 20 ms) to nearby and well-connected users in the area. Still, many real-time applications are sensitive to latency variations, e.g.,

*Both authors contributed equally to the paper

virtual reality, and some probes experience degraded performance due to long tails in their RTT distributions.

We further explore the network paths of the problematic probes in order to identify the source links for the large variations in RTT. The variation is likely to be located at congested links. The result from our exploration indicates that the problems appear predominantly on the cloud ingress link, ~ 65% of the time, even in the case of direct peering with Internet Service Providers (ISP).

To our best knowledge, this work is the first of a kind to evaluate cloud connectivity in favorable conditions at a large scale, concentrating on metropolitan areas. The main contributions of the paper are: (1) a pervasive quantification of RTT to 114 cloud datacenters located in 27 countries and 51 metropolitan areas, and (2) the detection and classification of the network links causing high RTT variations.

2 METHODOLOGY

In this section, we describe our measurement setup, tools, dataset, analysis methodology and discuss the limitations of our traceroute approach.

2.1 Areas of measurements

We select metropolitan areas with a population size over 1M which have at least one public cloud datacenter in the vicinity ($\leq$ 50 km from the city center). Thus, we measured 17 cities in Asia, two cities in Africa, 15 cities in Europe, 13 cities in North America, one city in South America, and three cities in Oceania. A listing of the targeted metropolitan areas can be seen in Figure 2.

2.2 Cloud End-Points

In each of the selected areas, we include all nearby datacenters, totaling 114 of them. We limit our choice to 10 cloud providers – Amazon, Google, Microsoft, IBM, Alibaba, Oracle, DigitalOcean, Linode, UpCloud, and Vultr. We picked these providers according to their proximity to our measurement areas and how well-established they are. The majority of the selected datacenters are also reached through Cloud Harmony[14], which hosts end-points at the clouds intended for performance evaluation from the outside. The datacenters of Vultr and Linode expose their own end-points for testing.

2.3 RIPE Vantage Points for active measurements

We use the well-known and established measurement platform RIPE Atlas [12]. It has a wide geographical coverage, and it has been used extensively for measuring network reachability, connectivity, and performance. The platform provides thousands of small hardware vantage points, called *probes*, connected to the Internet in a variety of installation environments, ranging from home networks to managed network cores. All probes we use have a wired (Ethernet) connection to the Internet. Users can instruct Atlas to perform active network measurements from selected probes using built-in tools or user-provided.

Some probes are hosted by cloud providers and network operators, mainly for monitoring their network reachability from outside [15] and performance evaluations. Since cloud-hosted probes are not relevant for quantifying users' cloud connectivity, we filtered them out by examining the AS:es they belong to and by employing user-defined tags [16] of the probes. From the remaining probes, we selected randomly up to 20 probes from each of the metropolitan areas. In total, more than 700 probes were used.

2.4 Experimental Setup

In our measurement campaign we use the built-in `paris-traceroute` [17] to measure RTT from a probe to a cloud end-point. We launch probes with an interval of 4 minutes to each nearby datacenter for a period of at least 24 hours. By using the TCP variant of `paris-traceroute`, which uses TCP `SYN` probing packets instead of ICMP, we avoid low-prioritization or blocking of ICMP on the reverse path. We use full-sized MTU packets of 1500 bytes and send three probe packets per hop. Additionally, we conduct a special experiment to compare the performance of 1500 vs. 48 bytes packets. All measurements were completed from April 2021 to May 2021.

2.5 Limitations of traceroute

As noticed in [18], classical `traceroute` [19] and ping are not the best possible tools to evaluate RTT and jitter, often resulting in an overestimation, especially the jitter. `Paris-traceroute` provides better accuracy than classical `traceroute` since it maintains the flow-ids to avoid differentiation between load balancing paths. There are additional limitations of classical `traceroute`, such as low prioritization of ICMP `Time exceeded` messages on the reverse path [20]. It should also be noted that when an ICMP Time Exceeded packet is sent back, only 8 bytes of the original payload are appended to the ICMP packet. Thus, this may induce asymmetry with only the uplink affected by increased packet size. For high bandwidth networks it becomes less of an issue. Also, the family of traceroutes can not detect Layer 2 artifacts, e.g., MPLS tunnels [21], frequently used by cloud networks.

2.6 Detection of problematic links

Some of the network paths exhibit larger RTT variations to a cloud than other probes to the same cloud end-point. We

here present how we localize the links responsible for the variations.

It is challenging to pinpoint variation anomalies using `traceroute` traces. One obstacle is the asymmetry of forward and reverse paths, and that `traceroute` exposes only the forward path. A problematic link can very well be on the reverse path and hence hidden. Also, reverse paths might differ among traceroute hops, which sometimes leads to a seemingly illogical situation when RTT of one hop in the middle of the path is higher than RTT of the destination. The `reverse-traceroute` [22] tool could potentially infer reverse paths, but its usage would require data from additional vantage points. Therefore, we identify the links responsible for high RTT variation by employing the following heuristics based on the methodology presented in [23, 24]. We consider for anomaly analysis every hop i for each the following condition applies $\Delta_i - median(\Delta_i) > (1.4826 * MAD)$, where $\Delta_i = RTT_{i+1} - RTT_i$ and MAD is a median absolute deviation of all Δ_i. Similarly to [23], we first check that RTT increase per hop is consistent along the path and set the threshold of 3 ms that an outlier must exceed compared to the median to be included in the analysis ([23] uses 10 ms, but we analyze lower RTTs).

For the problematic links, we look up their autonomous system (AS) number or check if it belongs to a cloud provider's network [25, 26]. We also retrieve the AS for the preceding link (router), so we know both ends of the problematic link. Then we categorize the link as *intra-AS* if both parts of the link belong to the same AS and as *inter-AS* otherwise. If they both belong to the targeted cloud, it is classified as *intra-cloud*. If it is the last hop, it is classified as *cloud ingress*. We also define the special case of *cloud ingress – direct cloud ingress* for those cases when the probe enters the cloud directly from its AS. Lastly, if it is the first hop, it is classified as *last mile*.

We exclude from the analysis traces containing stars, i.e., cases where some routers in the middle of the path did not reply to the `traceroute` probes.

3 RESULTS

In this section, we discuss the main finding of our measurements, providing a comprehensive analysis of RTT, its variation, and problematic links on the paths.

3.1 RTT Overview

We now look at all the RTT samples that we collected during our experiments. The full details of the RTT measurements are available in Appendix A, and in Fig. 1, we present an excerpt from it – the New York metropolitan area. There are 29K samples of 16 RIPE Atlas probes accessing DigitalOcean, Linode, and Vultr datacenters. Each line in Fig. 1 represents

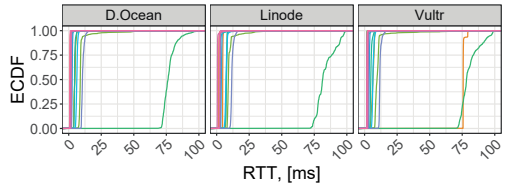


Figure 1: RTT measurements of the New York metropolitan area, each line represents individual RIPE Atlas probe (long tails cut, excerpt from Appendix A).

RTT distribution of an individual probe. What makes this case rather typical (see Appendix A) is that there are 14 well-connected probes having RTT below 10 ms and a couple of highly variant outlier probes suffering from the connectivity issues. As the mean RTT value of "good" probes is 4.3 ms (percentiles: 25^{th} – 1.87 ms, 50^{th} – 2.7 ms, 75^{th} – 6.36 ms, and 95^{th} – 11.24 ms), is it clear that in the New York area, it is possible to achieve RTTs below the motion-to-photon [13] threshold of 20 ms. This result is promising since it suggests that cloud datacenters deployed in metropolitan areas may run applications with stringent latency requirements, e.g., augmented reality. Although the distributions in Fig. 1 have long tails, the well-connected probes have only five outliers with RTTs around 100 ms and a single one over 1000 ms, which happened on cloud ingress.

Fig. 2 summarizes the results for all metropolitan area measurements. Generally, our considerations concerning the New York area apply also for most of the cities, where a few misbehaving probes increase the RTT variation. Despite that, the percentiles of RTTs for the entire dataset are as follows: 25^{th} – 2.99 ms, 50^{th} – 6.63 ms, 75^{th} – 15.93 ms, and 95^{th} – 36.6 ms. The high variance in RTT is observed in the city of Las Vegas because only three probes participated in measurements, of which one was highly unstable (see Appendix A). The notable exception is Salt Lake City, where RTTs fluctuate from 25 ms to 60 ms, with no single "good" probe. As our investigation shows, the remarkable slowdowns happen on cloud network ingress, inside local ISPs, and also on their boundaries. Moreover, some of the IPs inside the cloud network appear to be of Google's Canadian datacenters. Clearly, weaknesses or sub-optimal configuration of the networking infrastructure can eliminate the benefits of having a nearby datacenter. It is also likely that access is optimized for dedicated customers, not for the general public residing in the area. However, the case is not typical and maybe also due to the recent establishment of the datacenter [27].

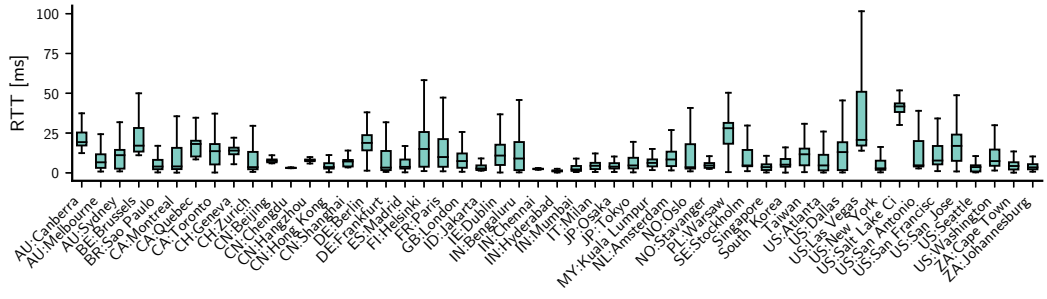


Figure 2: Summary of RTT distributions for individual cities and areas; see Appendix A for complete results.

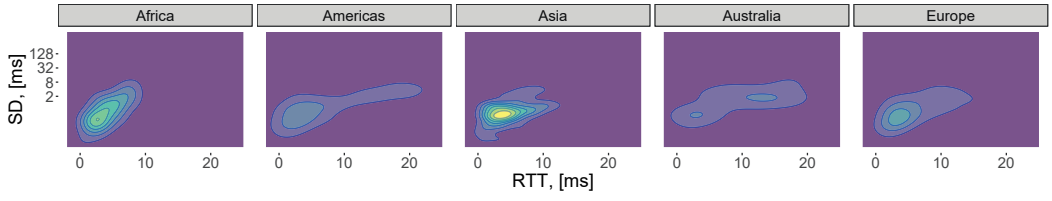


Figure 3: Clustering of probe-to-datacenter connections with respect to RTT and its SD grouped by continents.

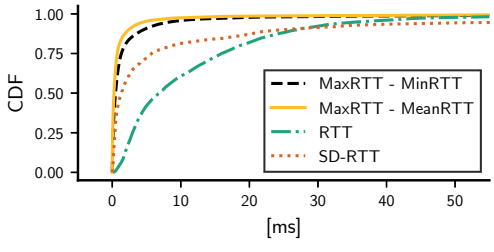
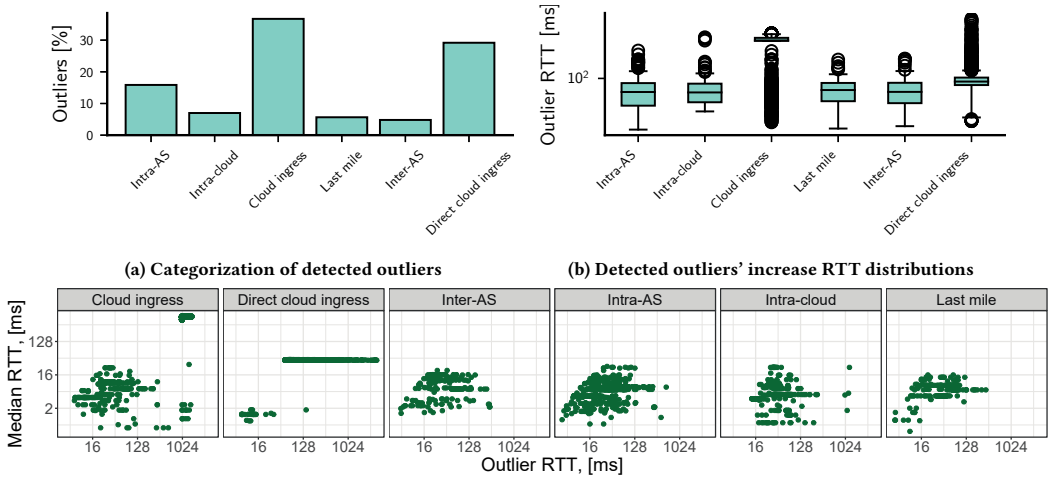


Figure 4: Comparison of RTT and RTT variation distributions expressed in terms of standard deviation (SD), difference between the maximum and minimum RTT, and difference between maximum RTT and mean RTT for every traceroute.

3.2 RTT Variation Overview

In this section, we further investigate the variation of RTT that we witnessed in our measurements. Fig. 3 shows a heatmap representing the RTT for a given path, matched against its standard deviation (SD) grouped by continents. In all cases, except for Australia, we distinguish clearly one cluster, which is especially distinctive and delimited by the yellow surface in the case of Asia. This cluster indicates that a significant fraction of the measured probe-to-datacenter connections has RTT value of ~ 5 ms and a SD of RTT lower than 2 ms. The percentiles of SD for the entire dataset are as follows: 25^{th} – 0.35 ms, 50^{th} – 1.23 ms, 75^{th} – 5.69 ms, and 95^{th} – 53.57 ms, and median absolute deviation is 6.85 ms.

We now analyze RTT variation from other perspectives, in addition to the SD. Fig. 4 show the cumulative distribution of different ways of representing RTT variation and also reports how these compare to the distribution of (all) RTTs, shown in dashed-dotted green, and its relative SD, shown in dotted red. From the figure, we see that 80% of the SD values are between ~0 ms and 10 ms, and ~ 50% is below 2 ms. Then, we propose two other ways to interpret RTT variation. The first is the range of the RTTs returned from the final hop of each `traceroute`, which can be calculated as the difference between the maximum RTT and the minimum RTT; this is shown as the dashed black curve in the figure. The second is the distance between the maximum RTT and the mean RTT of the final hop returned by each `traceroute`; this is indicated by the yellow line in the figure. The distributions of these two sets of values (black and yellow lines) are very similar, and the distance between them is barely distinguishable. This means that the span of the RTT values in metropolitan areas is very limited, exhibiting very low RTT variation; this



(a) Categorization of detected outliers

(b) Detected outliers' increase RTT distributions

(c) Anomalies by their type. Y-axis displays median RTT measured for the hop in "normal" conditions, x-axis – RTT in the case of anomaly.

Figure 5: Analysis of the detected outliers.

is also remarked by the fact that the 95th percentile of these distributions is below 10 ms.

3.3 Problematic links

Using the methodology described in § 2.6, we analyze the problematic links, causing what we call outliers, identified in our network measurements. In Figure 5a, we present the amount, expressed in percentage, of outliers grouped by the network location where they occurred. The outliers' dataset contains 3985 data points in total. Anomalies happening inside ASes (*Intra-AS*) and between them (*Inter-AS*) are 16% and 5%, respectively. In accordance with [28], the last-mile link (from the probe to the ISP's network) is one of the main responsible for causing cloud unreachability. In our case, the last-mile link is responsible for 5.6% of the detected outliers (*First Hop*). Typically, probes with a constantly high variation of RTT suffer from this problem. Around 36% of RTT outliers happen on the cloud ingress (*Cloud Ingress*), which is a relatively high number given that cloud providers constantly seek to optimize their peering agreements for better customer service. It is worth mentioning that often the last-mile (ISP's network) leads directly to the cloud network [6]. In the case of direct peering (*Direct cloud ingress*), the identified outliers account for 29% of the total outliers. Then, cloud networks (*Intra-cloud*) are responsible for 7% of the outliers.

Figure 5b quantifies the magnitude of the outliers', reporting their respective RTTs. Thus, the figure shows the distributions, grouped by network locations, of the outliers' RTTs. Here we see that the distribution of outliers in *Inter-AS* has a median RTT of 59 ms, while outliers within the AS, *Intra-AS*, account for a median RTT of 54 ms. Additionally, outliers at the *Cloud Ingress*, without direct peering, are in the order of 945 ms; here we have one probe, likely with connectivity issues, that delivers RTTs in the orders of seconds and skews the distribution. Outliers with direct cloud peering, *Direct cloud ingress*, have a median RTT of around 160 ms. According to this result, direct peering between ISPs and cloud providers seems to deliver better network performance in case of anomalies. Outliers within the cloud network, *Intra-Cloud*, experience increases of median RTT of 68 ms while outliers in the last-mile (*First Hop*) have a median RTT of 55 ms.

Finally, we show in Figure 5c a more granular representation of the detected outliers. On the y-axis, we represent the median RTT that a particular hop usually experiences. Then, on the x-axis, we report the extent of the RTT identified in the outlier under investigation. Please notice the non-linear scale used on both axis to improve visibility. Here it is possible to see how much an outlier deviates from the median RTT and, the more the markers deviate from the logarithmic curve on the right side, the bigger the value of the outlier.

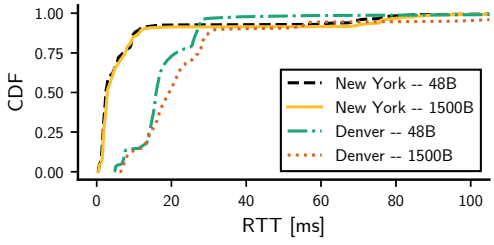


Figure 6: Comparison of RTT distributions for datacenters in New York and Denver for different packet sizes (48 and 1500 bytes).

3.4 Use Case: Packet Size and Distance

In this section, we investigate the impact of different packet sizes of the `traceroute` probes sent to the datacenters. We consider packet sizes of 48 and 1500 bytes, which is the maximum MTU size to avoid fragmentation. In this small-scale experiment, we want to see whether the probes' packet size has an impact on the RTT performance when combined with distance to datacenters. Hence, we showcase the situation in New York, where the Atlas probes are nearby (10–25 km), and in Denver, where the Atlas probes are located ~150 km away from the datacenters. In New York, the RTT distribution obtained with probes of size 48B is comparable to the one obtained with probes of size 1500B. From this, we deduce that the impact of "small" or "big" packets is negligible when the datacenters are close by. The 50^{th} percentile is below 10 ms, and the 90^{th} is below 15 ms. On the other hand, the situation in Denver is different. In fact, it is now possible to see how the RTT performance degrades when issuing probes with bigger packet size (1500B) and how the distribution obtained by 48B packets delivers better RTT performance. The 95^{th} percentile for 48B packets is 27 ms and for 1500B is 70 ms. Thus, we infer that the packet size of the probes does not particularly impact RTT performance when the vantage points are very close to the datacenters (10–25 km), however, when the distance to the datacenters increases, the size of the packet is accountable for the loss of performance.

4 RELATED WORK

In recent years, a number of cloud reachability studies were presented, demonstrating a great interest in the topic. The two closest works to ours are by Mohan et al. [9] and Corneo et al. [11]. These studies reflect how much had changed since the early days of the cloud when the delay to the cloud varied from hundreds of milliseconds to seconds [29]. In [9], the authors question the liability of latency reduction as a driver for edge computing development, noticing that RTTs to the cloud has reduced dramatically in the last decade. In [11], the authors assess the cloud reachability on a global scale, summarizing by continents, individual countries, and statistical areas, concluding that in half of the world's countries, RTT to the cloud stays within 20 ms to 40 ms range. Compared to the two studies above, our investigation has a much stronger focus on measurement locations, as we choose large metropolitan areas where the majority of the population resides, and network connectivity is generally good. Thus, we highlight the best possible performance that the cloud can offer in densely populated areas, where most of the demand for latency-critical applications is concentrated. Agrawal et al. investigate the *unreachability* of cloud services that they found to depend not on the cloud network but on the last-mile links [28]. We also acknowledge the last-mile connectivity problems, as in every city, there is a minority of probes experiencing problems related to their local ISP.

The following work is loosely related to the reachability topic but is still relevant for our research. Arnold et al. assessed the flattening of the Internets by showing that several cloud providers bypass Tier 1 ISPs [6]. Their methodology involves issuing ICMP `traceroutes` from virtual machines from six different cloud providers to wide IPv4 prefix ranges. Another work by Arnold et al. [30] evaluates the effects of private WAN on cloud performance. Palumbo et al. evaluate the latency performance of globally spread Amazon Web Services and Microsoft Azure datacenters from 25 vantage points from PlanetLab [31]. Haq et al. study inter-continental links between three major cloud providers and find them to deliver better RTTs and jitter when compared to public Internet links [32]. Tomanek et al. evaluate the latency performance of Microsoft Azure datacenter adopting a multidimensional approach [33]. Høiland-Jørgensen et al. study latency variation at Internet scale from existing datasets [34].

Our work is the first of a kind to explicitly evaluate cloud connectivity in favorable conditions at a large scale, concentrating on the areas where most of the existing and potential clients reside. Furthermore, we focus particularly on latency variation, which is one of the timing requirements of latency-sensitive applications reported by the 3GPP [35].

5 CONCLUSION AND FUTURE WORK

In this paper, we evaluated connectivity to the public cloud in large metropolitan areas on a global scale. We were motivated to investigate whether the public cloud can support novel latency-stringent applications in conditions where geographical distance has a very limited impact on performance. Our results have revealed the strong performance of the network, as for the entire dataset, the median RTT is 6.49 ms, and for the half of the probe-to-datacenter connections, standard deviation of RTT is below 1.23 ms. These results indicate

that the network latency in metropolitan areas is abundantly below the motion-to-photon threshold of 20 ms that latency demanding applications such as augmented reality require. We also have shown that remarkably often, delays happen at cloud ingress (~ 65%) and within the same AS (~ 16%).

The expansion of cloud providers is continuing actively, and major companies, e.g., Amazon, Microsoft, etc., are constantly increasing their presence in more and more regions around the world. The study of cloud connectivity in urban areas was also important because more than half of the world's population lives in cities, and the trend is likely to progress further.

We leave to future works the extension of our measurements to additional locations, e.g., datacenters in smaller regional cities, and to include in the study also non-global cloud providers, which are not currently addressed in this work. Moreover, we plan to complement our current methodology with techniques for reverse path examination and outliers detection. We also aim to include in future datasets wireless vantage points, e.g., Wi-Fi and 5G.

REFERENCES

[1] A. Fox *et al.*, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[2] Gartner, "Gartner Forecasts Worldwide Public Cloud Revenue to Grow 6.3% in 2020," https://www.gartner.com/en/newsroom/press-releases/2020-07-23-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-6point3-percent-in-2020, 2020.

[3] ——, "Gartner Forecasts Worldwide Public Cloud End-User Spending to Grow 18% in 2021," https://www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in-2021, 2020.

[4] A. W. Services, "Global infrastructure," https://aws.amazon.com/about-aws/global-infrastructure/, 2021.

[5] Microsoft, "Azure Geographies," "https://azure.microsoft.com/en-us/global-infrastructure/geographies/", May 2021.

[6] T. Arnold *et al.*, "Cloud provider connectivity in the flat internet," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 230–246. [Online]. Available: https://doi.org/10.1145/3419394.3423613

[7] M. Satyanarayanan *et al.*, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[8] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[9] N. Mohan *et al.*, "Pruning edge research with latency shears," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, 2020, pp. 182–189.

[10] United Nations, "World urbanisation prospects," "https://population.un.org/wup/DataQuery/", May 2021.

[11] L. Corneo *et al.*, "Surrounded by the clouds," 2021.

[12] RIPE NCC., "RIPE Atlas," "https://atlas.ripe.net/", 2020.

[13] K. Mania *et al.*, "Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity," in *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, 2004, pp. 39–47.

[14] CloudHarmony, "CloudHarmony – Transparency For the Cloud," "https://cloudharmony.com/", May 2021.

[15] V. Bajpai *et al.*, "Lessons learned from using the ripe atlas platform for measurement research," *ACM SIGCOMM CCR*, vol. 45, no. 3, 2015.

[16] RIPE NCC., "Probe tags," "https://atlas.ripe.net/docs/probe-tags/", 2020.

[17] B. Augustin *et al.*, "Avoiding traceroute anomalies with paris traceroute," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 153–158. [Online]. Available: https://doi.org/10.1145/1177080.1177100

[18] C. Pelsser *et al.*, "From paris to tokyo: On the suitability of ping to measure latency," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 427–432.

[19] V. Jacobson, "Traceroute," "ftp://ftp.ee.lbl.gov/traceroute.tar.gz", February 1989.

[20] NetBrain, "Traceroute limitations explained," "https://www.netbraintech.com/blog/limitations-of-traceroute/, 2020.

[21] B. Donnet *et al.*, "Revealing mpls tunnels obscured from traceroute," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 2, pp. 87–93, 2012.

[22] E. Katz-Bassett *et al.*, "Reverse traceroute." in *NSDI*, vol. 10, 2010, pp. 219–234.

[23] P. Raman *et al.*, "Building out the basics with hoplets," in *Passive and Active Measurement*, O. Hohlfeld *et al.*, Eds. Cham: Springer International Publishing, 2021, pp. 355–370.

[24] R. Fontugne *et al.*, "Pinpointing delay and forwarding anomalies using large-scale traceroute measurements," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 15–28.

[25] IPData, "IP Geolocation and Proxy Detection API," "https://ipdata.co", May 2021.

[26] MyIP, "MyIP," "https://myip.ms", May 2021.

[27] Ron Miller, "Google Cloud's newest data center opens in Salt Lake City," "https://techcrunch.com/2020/02/27/google-clouds-newest-data-center-opens-in-salt-lake-city", February 2020.

[28] K. Agrawal *et al.*, "Monitoring cloud service unreachability at scale," in *IEEE INFOCOM 2021*, 2021.

[29] A. Li *et al.*, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 1–14.

[30] T. Arnold *et al.*, "(how much) does a private wan improve cloud performance?" in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 79–88.

[31] F. Palumbo *et al.*, "Characterizing cloud-to-user latency as perceived by aws and azure users spread over the globe," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[32] O. Haq *et al.*, "Measuring and improving the reliability of wide-area cloud paths," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 253–262. [Online]. Available: https://doi.org/10.1145/3038912.3052560

[33] O. Tomanek *et al.*, "Multidimensional cloud latency monitoring and evaluation," *Comput. Netw.*, vol. 107, no. P1, p. 104–120, Oct. 2016. [Online]. Available: https://doi.org/10.1016/j.comnet.2016.06.011

[34] T. Høiland-Jørgensen *et al.*, "Measuring latency variation in the internet," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 473–480.

[35] 3GPP, "Service requirements for cyber-physical control applications in vertical domains," 3rd Generation Partnership Project (3GPP), Technical Specification (TS), 2020, version 17.4.0.

A ROUND-TRIP TIMES BY LOCATIONS AND CLOUD PROVIDERS

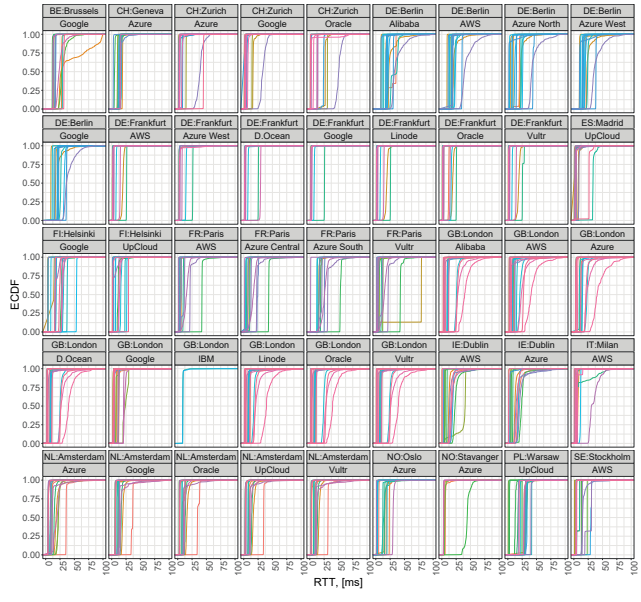


Figure 7: Europe: RTTs of individual probes by city and datacenter. Each line represents an individual probe. Long tails excluded.

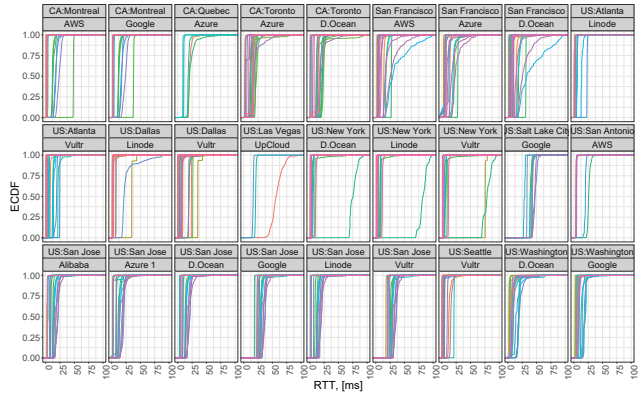


Figure 8: North America: RTTs of individual probes by city and datacenter. Each line represents an individual probe. Long tails excluded.

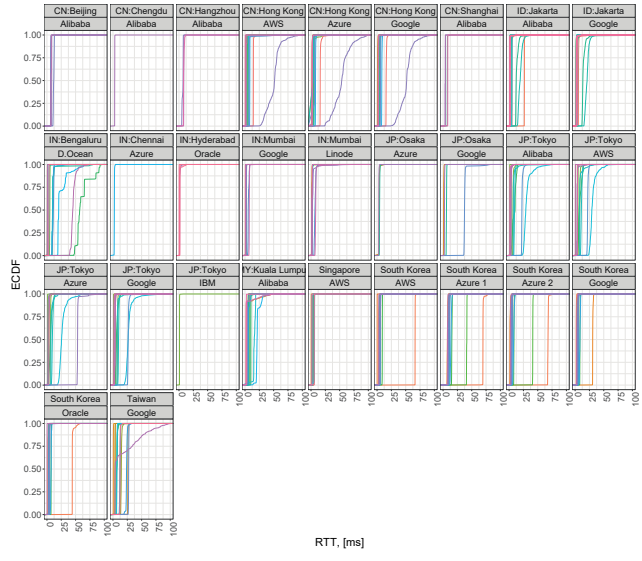


Figure 9: Asia, RTTs of individual probes by city and datacenter. Each line represents an individual probe. Long tails excluded.

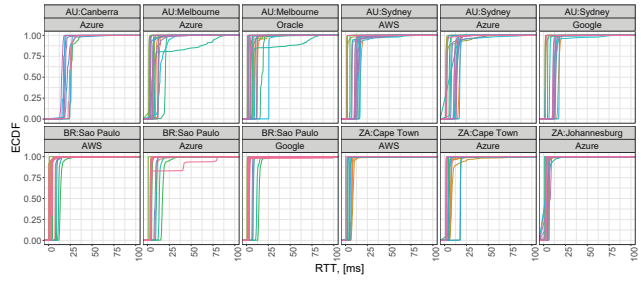


Figure 10: Australia, South America, and Africa: RTTs of individual probes by city and datacenter. Each line represents an individual probe. Long tails excluded.