Networked Latency Sensitive Applications - Performance Issues between Cloud and Edge

LORENZO CORNEO

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.

© Lorenzo Corneo 2021

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.


* 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.


Contents

Part I: Dissertation Summary ........................................................................................................ 13
1 Introduction ............................................................................................................................ 15
  1.1 Research Questions .......................................................................................................... 16
  1.2 Contributions .................................................................................................................. 18
  1.3 Methods .......................................................................................................................... 19
2 Edge Support for Sensing Devices ......................................................................................... 21
  2.1 Data Freshness ................................................................................................................ 21
  2.2 Cloud-Edge-Device Systems ......................................................................................... 22
3 Internet Measurements ........................................................................................................... 24
  3.1 The Internet .................................................................................................................... 24
  3.2 Tools ................................................................................................................................ 25
    3.2.1 Ping ........................................................................................................................... 25
    3.2.2 Traceroute ............................................................................................................... 25
  3.3 RIPE Atlas ........................................................................................................................ 26
  3.4 Cloud Computing Infrastructure ....................................................................................... 27
4 Edge Server Placement .......................................................................................................... 29
  4.1 Paths Discovery between the Edge and the Clouds .......................................................... 30
  4.2 Placement strategies ........................................................................................................ 31
5 Cloud Reachability ................................................................................................................ 33
  5.1 Latency Sensitive Applications ....................................................................................... 34
    5.1.1 Classification ............................................................................................................ 34
  5.2 Edge Computing Feasibility Zone .................................................................................... 35
  5.3 Latency Sensitive Applications in the Clouds ................................................................ 37
    5.3.1 General Cloud Reachability .................................................................................... 37
    5.3.2 Cloud Reachability in Metropolitan Areas ............................................................... 38
  5.4 Discussion of Results ....................................................................................................... 39
6 Summary of Papers ................................................................................................................ 40
7 Future Work ............................................................................................................................ 47
8 Conclusion ............................................................................................................................... 49
Svensk Sammanfattning ........................................................................................................... 51
Bibliography .............................................................................................................................. 53
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, Wenging 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-Ake 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.

Information 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:

$$\Delta(t) = t - U(t)$$  \hspace{1cm} (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 \( \tau_\Delta \), 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:

$$\bar{\Delta}(t) = \frac{1}{T} \int_0^T \Delta(t) \, dt$$  \hspace{1cm} (2.2)
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
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
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
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
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.
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 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 traceroutes 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,
~ 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, ~ 13 ms are taken by the display technology to refresh its content and this leaves a budget of ~ 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 ~ 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 ~ 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 1 - 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 wear-
able, 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 applica-
tions 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 up-
dates 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 so-
lutions. 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, Dig-
tal 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 super-
set 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 Amer-
ica. 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
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-
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-
tively. We quantified the end-to-end distance, as hop count, between users and cloud datacenters via \texttt{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 \texttt{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 \texttt{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 ≤ 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


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

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

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
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 pre-processed data for the traceroute measurements that he also collected.

Paper V

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. Consequently, 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

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 methodology 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


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.


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 placeringsalgoritms analserar olika placeringsalternativa 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


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


[42] Tan Bacinoglu and Elif Uysal-Biyikoglu. Scheduling status updates to


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

[56] man7.org. ping(8).
[65] RIPE NCC. RIPE atlas api reference.
[66] RIPE-NCC. Python client for ripe atlas api.
[67] RIPE-NCC. Official command line for ripe atlas.
[68] AWS. AWS Direct Connect Partners.
  "https://aws.amazon.com/directconnect/partners/".
[71] Lili Qiu, Venkata N Padmanabhan, and Geoffrey M Voelker. On the placement


[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*.


[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.


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”.)