TESI DI DOTTORATO

UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

DIPARTIMENTO DI INGEGNERIA ELETTRICA
E DELLE TECNOLOGIE DELL'INFORMAZIONE

DOTTORATO DI RICERCA IN
INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

CLOUD AND MOBILE INFRASTRUCTURE MONITORING FOR LATENCY AND BANDWIDTH SENSITIVE APPLICATIONS

FABIO PALUMBO

Il Coordinatore del Corso di Dottorato	Il Tutore
Ch.mo Prof. Daniele RICCIO	Ch.mo Prof. Antonio PESCAPÉ

A. A. 2019–2020

“To my family”

Contents

Acknowledgments **ii**

List of Figures **v**

Summary **viii**

1 Introduction and Background **1**

1.1 Features and QoS requirements of multimedia and telemedicine applications 1

1.2 Cloud, mobile cloud and edge computing 6

1.2.1 Cloud computing 6

1.2.2 Mobile cloud computing 10

1.2.3 Fog/edge computing 14

1.3 Thesis contributions 18

2 Cloud-to-user network latency characterization and application **23**

2.1 Latency of Internet paths 27

2.2 Latency monitoring in cloud networks 28

2.2.1 Monitoring latency in intra-datacenter networks (intra-DC) . . . 30
2.2.2 Monitoring latency in cloud WANs (inter-DC and C2U) 32
2.3 Experimental methodology . . . 36
2.3.1 Data collection . . . 36
2.3.2 Statistical tests for evaluation . . . 39
2.4 Experimental results . . . 40
2.4.1 Characterization of spatial and temporal latency trends 41
2.4.2 Impact of probing methods . . . 46
2.4.3 Hourly, daily and weekly patterns . . . 49
2.5 Application scenarios . . . 53
2.5.1 Badness events detection . . . 54
2.5.2 Multi-cloud deployments . . . 64
2.6 Remarks and discussion . . . 65

3 Active available bandwidth estimation in mobile networks 68
3.1 Background . . . 72
3.2 Reviewing bandwidth estimation using active approaches . . . 75
3.3 Methodology and preliminary experiments . . . 79
3.4 Experimental results . . . 84
3.4.1 Available bandwidth vs. achievable throughput estimates 85
3.4.2 Generated traffic volume . . . 88
3.4.3 Measurement latency . . . 90
3.4.4 Measurement latency vs. accuracy . . . 93
3.4.5 Evaluating setup impact . . . 95
3.4.6 Whole-day experiments . . . 98

3.5 Remarks and discussion 101

4 Passive bandwidth estimation in mobile networks leveraging SDN 103

4.1 Background 107

4.2 Methodology and setup 109

4.2.1 Measurement methodology 110

4.2.2 Experiment design 112

4.2.3 Measurement scenario 113

4.3 Experimental results 115

4.3.1 Results discussion 120

4.4 Remarks and discussion 121

Conclusions 124

List of Figures

1.1 Example of telepathology application for teaching purposes. . 4

1.2 Edge computing application for telepathology at Saint Louis University, USA. 5

1.3 NIST Cloud Computing reference architecture (source [70]). . 7

1.4 Mobile traffic growth and expected increase by 2026 (from Ericsson Mobility Report data and forecasts for 2020). 11

1.5 Cloudlets concept (source [104]). 15

2.1 Contributions to packet delay. 27

2.2 Geographical distribution of VPs and CRs in the experimental campaign. 37

2.3 Average latency (ms) (14-day span, TCP probing method, port 80). 42

2.3 Average latency (ms) (14-day span, TCP probing method, port 80). 43

2.4 Variability in terms of $D_{95-5} = 95^{th}pctl - 5^{th}pctl$. 44

2.5 Number of (dis)agreements between pairs of different probing methods about best provider across different protocols in the four CRs. 48

2.6 Evolution of mean latency values (in ms) considering hourly patterns. 50
2.7 Evolution of mean latency values (in ms) considering daily patterns. 51
2.8 Evolution of mean latency values (in ms) considering weekly patterns. 52
2.9 Badness % in $30\,\mathrm{min}$ buckets (14-day span, TCP probing method, port 80). 54
2.10 Persistence of badness events for both providers. 58
2.11 ECDF reporting the agreement percentage in detecting badness events across probing-methods pairs. 60
2.12 Online badness events at different aggregation. 62
2.13 Gains achievable with multi-cloud deployments. 65

3.1 Results of the preliminary tests involving Yaz and Pathload. The difference between available bandwidth and achievable throughput depends on the country. Pathload shows higher variability compared to Yaz. 82
3.2 Comparison between Available Bandwidth and TCP Achievable Throughput measurements, averaging over the 4 runs. While the specific relationship between available bandwidth and achievable throughput depends on the country, it is evident that available bandwidth can be used to approximate throughput. 84
3.3 Time series for TCP achievable throughput. 85

3.4 CDFs reporting the difference between TCP achievable throughput and Available Bandwidth estimates ((a) Relative and (b) Absolute) in the 4 different countries. 87
3.5 Comparison between traffic volume generated for Available Bandwidth and Achievable Throughput estimations. 90
3.6 Joint density distribution considering results versus measurements latency for available bandwidth estimation in Spain. . . 91
3.7 Available bandwidth estimates boxplots, mean (grey dotted line), and average TCP throughput (red dashed line) considering different run duration. 93
3.8 Distribution of differences between local and target spacings. . 97
3.9 Available bandwidth and TCP achievable throughput considering different duration for measurement intervals during the additional, whole-day experiments. 99

4.1 Details about the measurement setups considered for SDN-based passive bandwidth estimation. 107
4.2 Architecture of Software Defined Networks (source [72]). . . . 108
4.3 Probability density function (PDF) for the relative error in the wired-LAN deployment, considering local and remote controller deployment and different polling periods. 116
4.4 Relative error with different requested bitrates, in the case of a local and remote controller in the RAN deployment. 117
4.5 Relative error with variable polling periods in the case of a local and remote controller in the RAN deployment. 118

Summary

This PhD thesis involves the study of cloud computing infrastructures (from the networking perspective) to assess the feasibility of applications gaining increasing popularity over recent years, including multimedia and telemedicine applications, demanding low, bounded latency and sufficient bandwidth. I also focus on the case of telemedicine, where remote imaging applications (for example, telepathology or telesurgery) need to achieve a low and stable latency for the remote transmission of images, and also for the remote control of such equipment. Another important use case for telemedicine is denoted as remote computation, which involves the offloading of image processing to help diagnosis; also in this case, bandwidth and latency requirements should be enforced to ensure timely results, although they are less strict compared to the previous scenario.

Nowadays, the capability of gaining access to IT resources in a rapid and on-demand fashion, according to a pay-as-you-go model, has made the cloud computing a key-enabler for innovative multimedia and telemedicine services. However, the partial obscurity of cloud performance, and also security concerns are still hindering the adoption of cloud infrastructure. To ensure that the requirements of applications running on the cloud are satisfied, there is

the need to design and evaluate proper methodologies, according to the metric of interest. Moreover, some kinds of applications have specific requirements that cannot be satisfied by the current cloud infrastructure. In particular, since the cloud computing involves communication to remote servers, two problems arise: firstly, the core network infrastructure can be overloaded, considering the massive amount of data that has to flow through it to allow clients to reach the datacenters; secondly, the latency resulting from this remote interaction between clients and servers is increased. For these, and many other cases also beyond the field of telemedicine, the Edge and Fog computing paradigms were introduced. In these new paradigms, the IT resources are deployed not only in the core cloud datacenters, but also at the edge of the network, either in the telecom operator access network or even leveraging other users' devices. The proximity of resources to end-users allows to alleviate the burden on the core network and at the same time to reduce latency towards users. Indeed, the latency from users to remote cloud datacenters encompasses delays from the access and core networks, as well as the intra-datacenter delay. Therefore, this latency is expected to be higher than that required to interconnect users to edge servers, which in the envisioned paradigm are deployed in the access network, that is, nearby final users [120]. Therefore, the edge latency is expected to be reduced to only a portion of the overall cloud delay. Moreover, the edge and central resources can be used in conjunction, and therefore attention to core cloud monitoring is of capital importance even when edge architectures will have a widespread adoption, which is not the case yet. While a lot of research work has been presented for monitoring several network-related metrics, such as bandwidth, latency, jitter and packet loss, less attention was given to the monitoring of latency in cloud and edge cloud infrastructures. In detail, while

some works [116] target cloud-latency monitoring, the evaluation is lacking a fine-grained analysis of latency considering spatial and temporal trends. Furthermore, the widespread adoption of mobile devices, and the Internet of Things paradigm further accelerate the shift towards the cloud paradigm for the additional benefits it can provide in this context, allowing energy savings and augmenting the computation capabilities of these devices, creating a new scenario denoted as mobile cloud. This scenario poses additional challenges for its bandwidth constraints, accentuating the need for tailored methodologies that can ensure that the crucial requirements of the aforementioned applications can be met by the current infrastructure. In this sense, there is still a gap of works monitoring bandwidth-related metrics in mobile networks, especially when performing in-the-wild assessment targeting actual mobile networks and operators. Moreover, even the few works testing real scenarios typically consider only one provider in one country for a limited period of time [85, 103], lacking an in-depth assessment of bandwidth variability over space and time.

In this thesis, I therefore consider monitoring methodologies for challenging scenarios, focusing on latency perceived by customers of public cloud providers, and bandwidth in mobile broadband networks. Indeed, as described, achieving low latency is a critical requirement for core cloud infrastructures, while providing enough bandwidth is still challenging in mobile networks compared to wired settings, even with the adoption of 4G mobile broadband networks, expecting to overcome this issue only with the widespread availability of 5G connections (with half of total traffic expected to come from 5G networks by 2026).

Therefore, in the research activities carried on during my PhD, I focused on monitoring latency and bandwidth on cloud and mobile infrastructures, assess-

ing to which extent the current public cloud infrastructure and mobile network make multimedia and telemedicine applications (as well as others having similar requirements) feasible. In detail, the contributions presented in this thesis are manifold:

- I have measured latency from the two main public cloud providers as of today, namely AWS and Azure, during a 14 days experimental campaign. Consequently, I have conducted an in-depth characterization of cloud-to-user network latency, highlighting possible applications concerning detection and troubleshooting of abnormal situation, and evidencing the benefits of multi-cloud deployments.

- I have conducted an extensive campaign leveraging commercial mobile broadband networks (3G/4G) supported by the MONROE platform, to assess bandwidth leveraging active estimation methods, necessary on uncontrolled network paths. In detail, I have assessed the use of available bandwidth metric (at network layer) as a proxy for transport-layer achievable throughput, at a fraction of the probe (syntethic) traffic volume compared to achievable throughput measurements, and able to produce accurate and quick estimates.

- Consequently, I have evaluated a passive state-of-the-art method for bandwidth estimation leveraging a Software Defined Network approach. Considering again the MONROE platform, I have assessed the accuracy of such method in-the-wild and characterized several factors that can possibly impact it, including traffic rate, polling period and SDN controller deployment with respect to the switch.

These contributions reflect into the thesis structure: considering the next

Chapters, I first provide in Chapter 1 a background on the requirements of multimedia and telemedicine applications, then highlighting how the characteristics of cloud infrastructures and its variations can provide benefits to these applications, but also focusing on possible obstacles limiting its adoption and that thus require active investigation. Subsequently, in Chapter 2 I present an extensive characterization of network latency as experienced by users of cloud applications (denoted as cloud-to-user, or C2U, latency), while in Chapter 3 I assess active available bandwidth estimation in mobile broadband networks. The aforementioned SDN-based, passive approach is instead presented and discussed in Chapter 4. Finally, in the Conclusion Chapter I provide a final discussion and trace possible opportunities for future work and investigation.

Chapter 1

Introduction and Background

In this chapter I provide general background and context to better understand the contributions of the experimental work conducted and discussed in this thesis. First, in Sec. 1.1 I provide an overview of multimedia and telemedicine applications, focusing on their Quality of Service (QoS) requirements. Then, in Sec. 1.2 I discuss cloud computing infrastructure and its evolution and extensions over time, and how the aforementioned applications can benefit from the adoption of these different paradigms. Finally, Sec. 1.3 details the challenges that motivated the work in this thesis and the contributions that I present.

1.1 Features and QoS requirements of multimedia and telemedicine applications

With the rapid growth of information and communication technologies, multimedia applications, such as real-time video streaming, conferencing, video on-demand, have seen a surge in popularity [5]. They all have in common

bandwidth requirements, as video and image streaming at high quality requires a significant amount of bandwidth [5]. In addition, real-time video streaming services also have strict latency requirements, while other applications need only one QoS metric to focus on. The latency requirements vary with the specific class of applications, with real-time interactive ones being the most demanding. Woods et al. [126] highlight that typically the time required by a human being to respond to a stimulus is around 230 ms. However, these requirements can vary according to the application, as there are example requiring latency below 100 ms [78], or even stricter, as in the case of online gaming [20]. Cloud gaming poses as a high-demanding application since it requires both high-bandwidth for video streaming and low latency to ensure interactivity. In this case, it is also interesting to note that latency requirements vary with the individual ability, but values above 50 ms are clearly perceived by most users [98].

In addition to these, future applications made possible by the rising capabilities of mobile networks also include AR/VR, autonomous vehicles and smart cities services, where extremely low latency and high bandwidth are necessary, making them some of the most demanding examples [78], even requiring latency lower than 20 ms.

Similarly, several innovations were made possible also in the field of medicine, allowing a whole new range of applications known as telemedicine. Commonly, the term telemedicine refers to the delivery of healthcare and related services over long distances using communication technologies, in order to allow information exchange for diagnosis, treatment, and prevention of disease, but also for scientific research and for teaching purposes [57]. For example, patients in remote, rural areas can interact with a telemedicine system,

providing their personal information and reporting symptoms. In this case, the information are sent to a remote server, where an automated process assigns the patient to a doctor, who can then provide a prescription or a diagnosis to the patient remotely. The amount of computing and storage resources to deploy to guarantee high availability and adequate response times are challenging, especially when deploying these healthcare services leveraging dedicated physical systems. These issues perfectly fit the advantages provided by the adoption of cloud computing, which has therefore received significant attention in literature [43]. According to this analysis, the typical use cases of telemedicine fall within 6 categories [43]:

- Telemedicine/Teleconsultation;
- medical imaging;
- public health and patients' self-management;
- hospital management/clinical information systems;
- therapy;
- secondary use of data.

According to the literature surveyed in the aforementioned analysis, the first two are the most widespread applications, and indeed, the importance of teleimaging based applications in medicine, such as telepathology and teleradiology, has already been recognized for more than two decades [81, 123], and they have received increasing attention recently [114], also thanks to the emergence of new technologies and paradigms such as edge computing [102].

Figure 1.1: Example of telepathology application for teaching purposes realized by the University of Southern California, allowing students from Chattanooga to remotely access and control a 4K microscope (source [12]).

In detail, the telepathology application requires both consultation and computationally expensive image processing tasks performed on the images coming from slides under a microscope [6], and can be also leveraged for training and education purposes [50]. An example of this service is shown in Fig. 1.1, reporting the application for teaching purposes realized by Prof. Richard Weinberg at University of Southern California, to allow students from a high-school in Chattanooga to remotely access and control a 4K microscope deployed in the USC laboratories [12]. Similarly, another example of telepathology application is shown in Fig. 1.2, illustrating the work conducted at Saint Louis University and to which I have partially contributed during a visiting research period. In this application, a microscope is controlled remotely through a computer, which also acts as an edge server, able to execute processing tasks on the

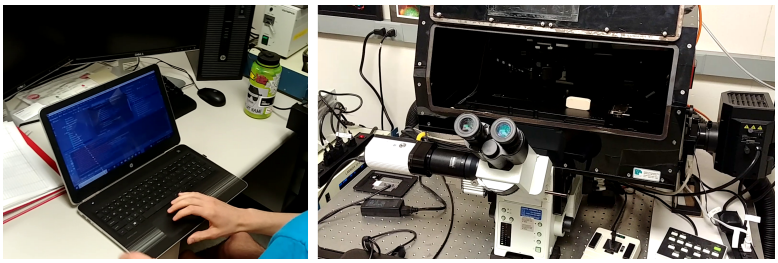


Figure 1.2: Edge computing application for telepathology at Saint Louis University, USA. A microscope is controlled remotely through a computer which also performs edge processing.

images coming from the slides, while remote pathologists and users can access the microscope through a web application [102]. As for multimedia applications, different telemedicine applications also have different requirements, often stricter given the critical scenario. For example, storage and retrieval of medical records is a less latency-sensitive application compared telepathology or telesurgery, that instead require real-time audio and video transmission. Indeed, telesurgery is negatively affected by low bandwidth, high delay, jitter and packet loss [1], at the point that latency can affect its entire feasibility. Similarly, telepathology also requires low, bounded latency and sufficient bandwidth to allow image streaming. In detail, in the case of the remote imaging use case, these requirements ensure that the microscope can be remotely controlled in an effective manner providing a good image quality. High response times impact on the capability of moving the slide and the microscope precisely, and low bandwidth compromises the image quality, possibly impacting on the diagnosis itself. Instead, when images need to be processed remotely (a use case also denoted as remote computation), low latency and high band-

width are needed to reduce the time between the offloading and the return of the computation results to the final user. This is a good example to remark that, according to the application, the latency requirement is also linked to the bandwidth, since transmission time of the application data is clearly influenced by link capacity and spare bandwidth. Nanda and Fernandes [83] focus on the QoS requirements of remote monitoring applications, enforcing congestion control, admission control, setting up virtual circuits and traffic differentiation at backbone network layer, with the adoption of data-link technologies such as ATM, DiffServ in Ethernet networks (by properly setting the DSCP values) and MPLS. These mechanisms allow to satisfy the high bandwidth and low response times requirements.

Of course, in addition to latency and bandwidth, it should be noted that (for example when storing health records), the communication has to deal with the security requirements and keep the data strictly confidential, and different solutions have been applied for this goal, covering network or higher levels of the protocol stack to provide security, for example by using IPSec [1].

Considering the requirements of these applications and as already mentioned briefly, the adoption of cloud infrastructures offers novel possibilities but also poses additional challenges, as detailed in the next section.

1.2 Cloud, mobile cloud and edge computing

1.2.1 Cloud computing

In 1961, Prof. John McCarthy said “Computing may someday be organized as a public utility just as the telephone system is a public utility”. This vision became reality with the advent of cloud computing, which has considerably

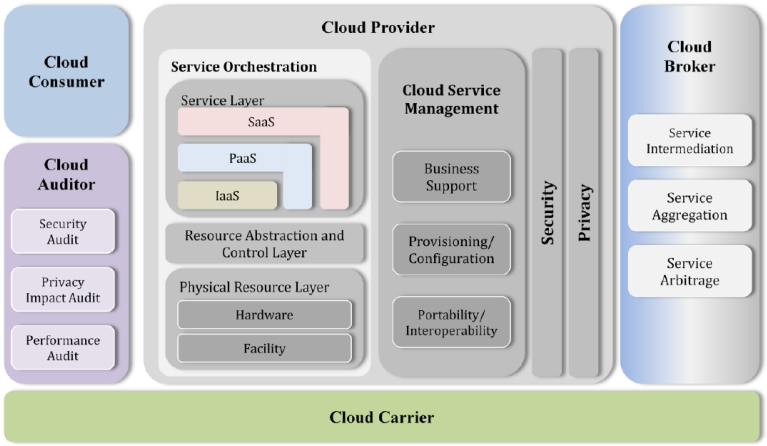


Figure 1.3: NIST Cloud Computing reference architecture (source [70]).

increased its popularity over the last decade, providing users with access to all sorts of resources over an Internet connection. According the National Institute of Standards and Technology (NIST), cloud computing is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". [73]. Through its adoption, capital expenditure is reduced, since the cost of buying, deploying and maintaining these resources on premise is partially or completely eliminated. In this way, users leverage computational, storage, network and several other types of resources using their Internet connection, whose performance therefore becomes a crucial element for the success of cloud applications.

NIST has also identified five essential characteristics, three service models, and four deployment models for cloud computing. The five key features are the following:

- **On-demand self-service**. This means that human interaction with the cloud provider should be minimal, and that customers can unilaterally provision the different resources provided as services.
- **Broad network access**. All the resources are accessed through a standardized mechanism requiring a network connection.
- **Resource pooling**. A cloud provider serves multiple customers according to a multi-tenant model, where physical resources are shared among multiple users by means of virtualization technologies.
- **Rapid elasticity**. The requested resource can scale dynamically up and down, even automatically, to adapt to user demands quickly.
- **Measured service**. In order to ensure the optimal use of resources, cloud providers have metering capability, and offer a certain degree of visibility to customers.

Following a utility model, cloud resources are rented according to a pay-as-you-go billing model, meaning that users are only charged for the actual time spent using their resources.

The overall reference architecture of Cloud computing as envisioned by NIST [70] is shown in Fig. 1.3. This figure also reports the three service models originally identified, which are defined according to the abstraction provided to the final user. Indeed, in the **Software as a Service** (SaaS) model, customers use a software remotely, without having to physically install it on their machine and having to comply with the software requirements. Typically, instead, developers leverage the **Platform as a Service** (PaaS) model to build and deploy their applications directly onto the cloud; in this case, cloud ser-

vices provide a platform for software development and deployment using cloud resources. Access to virtualized resources, such as virtual machines or storage, can instead be acquired through the **Infrastructure as a Service** (IaaS) model. This model is therefore mostly targeted at system administrators or users needing a lower-layer abstraction of the cloud resources. In addition to these service models, several others were proposed over the years, like **Sensing-as-a-service** or models in the Internet of Things (and denoted as Cloud of Things) context [130].

Starting from these service models, customers of a cloud provider can become service providers themselves, building their services on top of the cloud infrastructure and providing it to the final users. This kind of model is common and adopted by several services today; considering for example Netflix as one of the most notable example, leveraging the infrastructure provided by Amazon Web Services (AWS), including for instance their storage options (like S3), to deploy their Video On Demand service.

In addition to the service models, different deployments options also were identified in the original cloud definition, regarding the way access to cloud resources is given to external users. Specifically, cloud resources are commonly used by customers leveraging **public cloud providers**, which make these resources available to anyone via the renting model discussed before. Examples of cloud providers are AWS, Azure, Alibaba Cloud and Google Compute, which also detain together the highest market share as of today [37]. However, there are some alternative models; in some cases **private cloud** infrastructures are built, typically by big enterprises that, for security reasons, want their data to remain on their premises. In other cases multiple organizations with shared concerns deploy their own cloud infrastructure, in which case the **community**

cloud term is used. Combining these deployment options, **hybrid** deployment models are also adopted, that is, using different cloud infrastructures linked by (standardized or proprietary) technologies to enable data and application portability.

Summarizing, despite the service models and the deployment options, the access to resources (computational, storage, network) rented via the cloud infrastructure strongly relies on the performance of the Internet connection used to link users to them. With the growth of demand, it is crucial to ensure that user-perceived performances meet the *service-level agreements* (SLA) stipulated between providers and customers. However, there is also a partial lack of visibility into performance, as public cloud providers offer limited or qualitative information into the performance provided by their resources [93], resulting in a possible unpredictability of the performances of applications deployed on the cloud, both for network or provider related reasons. These add to other concerns partially hindering the adoption of public cloud infrastructures, such as privacy concerns or refrain from using vendor locked-in solutions. These aspects motivate the focus on non-cooperative monitoring approaches, allowing to gather visibility into cloud performance from an external, non-privileged point of view.

1.2.2 Mobile cloud computing

The surge in the adoption of mobile devices, such as smartphones and tablets, is evident as of today, and so is the volume of traffic generated by them, as highlighted by the latest Ericsson Mobility report for 2020 [33]. While the pandemic has slowed down the number of new subscriptions during 2020, the mobile traffic volume has steadily increased, as shown in Fig. 1.4, also re-

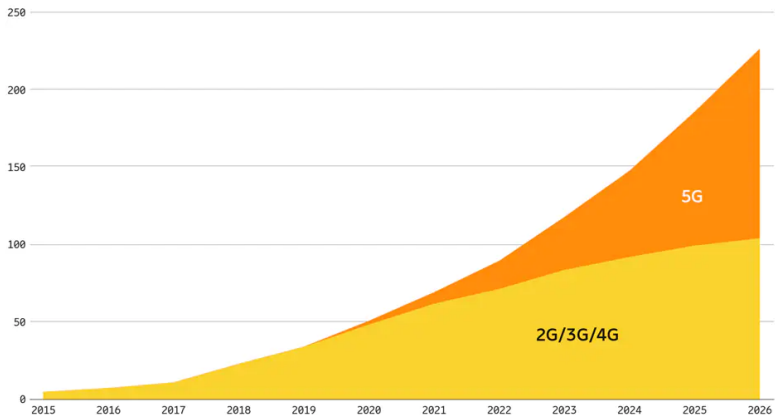


Figure 1.4: Mobile traffic growth and expected increase by 2026 (from Ericsson Mobility Report data and forecasts for 2020).

porting that by 2026 54% of the mobile traffic is expected to be related to 5G networks. Moreover, smartphones are responsible (in 95% of the cases) of this traffic, currently made up for 66% by video traffic, a share that is expected to increase to 77% by 2026.

While these devices are becoming widespread, the number and types of applications that can be run on them are limited by their computational resources (even in the case of expensive high-end devices) and from energy consumption constraints, as they are battery powered, and finally by reduced storage capabilities. These limitations have led to the emergence of the **mobile cloud computing** (or MCC) paradigm [124], which can be seen an extension of the cloud computing model where the device leveraging the cloud resources, typically according to the PaaS or SaaS models, is a mobile one, which offloads computational intensive tasks and the storage of massive amounts of data to the cloud.

This paradigm can bring several benefits, extending the battery life of the mobile devices, improving their storage and computational capabilities, and allowing to easily access data from multiple devices. Also, scalability is ensured, allowing services to adapt to users demands, while reliability and availability of data and application running in the cloud are improved compared to keeping them on mobile devices, since in this case they are more likely to be lost. The growing availability of mobile broadband technology, with 4G networks already reaching significant capabilities, and extended speeds and coverage expected by the 5G deployments with femtocells (expecting to contribute to more than a half of the mobile network traffic by 2026), will give increasing importance to this paradigm in the near future.

With the diffusion of mobile devices, especially with wireless sensors and IoT devices, the mobile cloud computing paradigm has naturally found his way into telemedicine applications as well. Indeed, while mobile devices are currently used for basic healthcare services, such as maintaining schedules, booking appointments, interacting with nurses and doctors, accessing their records, it is evident that several additional possibilities open up with the introduction of the cloud paradigm to overcome the limitations in computational capacity and power consumption. In detail, the integration of MCC and telemedicine provides the availability of health applications and medical information anywhere and anytime, and significantly higher computing resources for more complex monitoring and processing performed in real-time. For this reason, the importance of mobile cloud computing for telemedicine has recognized and described in previous works [39, 121]. For example, health monitoring tasks [40], or social assistive robotics can benefit from the adoption of mobile cloud computing, in this latter case leading to a novel service model denoted

as Robot-as-a-Service (Raas) [15].

While the potential benefits are evident, as pointed out by previous work [113, 121], one of the main obstacles for the widespread adoption of mobile cloud computing in telemedicine is that mobile users, sensors or IoT devices may experience congestion due to bandwidth congestion, network disconnection and signal attenuation, resulting in service interruption or delay in the communication with the cloud. Moreover, as mobile devices are operated on heterogeneous wireless networks, there can be large differences in the network bandwidth capacity and communication quality, further accentuated by the mobility aspect [113]. This poses once again serious attention to the bandwidth and delay requirements, given the critical applications related to healthcare, and motivates the implementation of accurate monitoring methodologies. These issues are often tackled by acting at the physical or data-link layers, for example (especially in the case of IoT devices) employing different transmission technologies [75]. Furthermore, the delay between the mobile device request and cloud service response can be significant, and is dependent on the distance between the device and the cloud infrastructure, as well as the processing capabilities on the Virtual Machines rented to host the telemedicine application. Therefore, several efforts were put into analyzing the public cloud infrastructures, as done for example by Haider et al. [48], who have characterized the performance of a telemedicine web service hosted on Amazon EC2 VMs under increasing workloads, assessing the conditions (in terms of requests throughput) in which the infrastructure is able to provide low, guaranteed response times to telemedicine applications.

Finally, I recall that data security, privacy and confidentiality are all key issues in this context. Although the study of security mechanisms for cloud

computing applied to telemedicine is not the main focus of this thesis, I have contributed to surveying the literature about Intrusion Detection Systems in the Mobile Cloud context, also highlighting open research issues [105]. In this case, indeed, there is a clear tradeoff between performance and security, accentuated by the heterogeneous and wireless scenarios, and by the multi-tenancy typical of cloud environments.

1.2.3 Fog/edge computing

In recent years, other trends have risen, with the goal of extending or complementing the core and mobile cloud computing paradigms, particularly to address the additional delay induced by the core cloud paradigm, constituting an issue for latency-sensitive applications. Indeed, the offloading and the communication with the core cloud introduce an additional delay, and also pose additional load into the core network, possibly leading to congestion, up to the point where the centralized model consisting of few large cloud datacenters will not be able to provide enough bandwidth for novel and widespread applications [12].

For these reasons, Satyanarayanan et al. [104] first proposed to deploy additional computational resources, denoted as cloudlets, closer to the end-users, that is at the *edge* of the network. In comparison with the core cloud, resources deployed in a cloudlet are more limited in order to reduce the infrastructural expenses, as this deployment is supposed to be much more capillar. The concept of cloudlet was then refined and expanded by Verbelen et al. [120], considering to divide the applications running on the cloudlets into functional components that can be individually executed at the edge network. The work also focuses on the deployment aspect of the computational resource nearby the

Figure 1.5: Cloudlets concept (source [104]).

mobile users, that can be for example colocated with the access point for WiFi networks or in the Base Station in the case of cellular networks. In this envisioned architecture, users can run custom Virtual Machines on the cloudlets, offloading their applications similarly to the core cloud paradigm, but with a reduced latency. This novel paradigm where computation is shifted closer to users (by means of cloudlets) is denoted as **fog/edge computing**, to remark its vicinity to end-users compared to cloud computing. I remark that while some works interchangeably use the term fog or edge computing, others separate the *edge* computing paradigm as one employing only local (edge) resources, and using the *fog* term when both cloud and local resources used in conjunction [51]. For example, the fog layer is often leveraged to perform preprocessing

operation and alleviate the burden on the core cloud, as done in [26] considering data coming from wireless sensors; in other cases a decision is made to execute tasks on the core or on the edge according to their requirements, aiming at maximizing resource usage while satisfying user needs. Comparing the latency experienced by final users in the cloud and edge paradigms, it can be seen that the overall latency experienced when connecting to remote cloud datacenters (that is, the cloud-to-user latency) includes propagation and transmission delays from the user access network, as well as the core network and the intra-datacenter network connecting the resources within the datacenter. Therefore, cloud-to-user latency is expected to be higher than the edge-to-user latency required to interconnect final users to their closest edge servers. Indeed, in the envisioned paradigm the edge resources are deployed in the access network (or within the base station in the case of mobile networks), and as such are placed nearby final users [120]. These considerations remark that, even if the edge computation time could be higher (since less resources are typically deployed at the edge to reduce costs), the overall edge-to-user latency is expected to be reduced to only a portion of the cloud-to-user delay.

Thus, the edge paradigm represents an opportunity for different application scenarios [106], for example in the smart home, smart city, video analytics domains; however, several new challenges also arise. Indeed, since less resources are deployed at the edge, the choice of which tasks offload to the edge and to the core cloud is not trivial and has attracted significant research efforts, proposing solutions on how to concretely realize the edge cloud [12].

It is clear that the adoption of fog/edge computing paradigm and the diffused deployment of cloudlets can provide benefits to telemedicine applications [57], by adding an intermediate layer between the mobile devices and the

remote cloud servers to reduce the response time and allowing task offloading and provisioning of computing resources. The recognized advantages of fog computing in healthcare are well highlighted in [107], focusing on distributed computing and offloading from mobile devices to support analytics while meeting the QoS requirements. Several kinds of application from those introduced in Sec. 1.1 can benefit from this paradigm; for example Gia et al. [40] leverage fog computing for feature extraction from ECG signals collected via IoT devices, achieving low response times and bandwidth efficiency. Compared to the other applications, telemedicine services leveraging the edge computing paradigm face similar challenges, for example regarding resource and VM placement, as discussed in [54], where authors focus on VM migration strategies in mobility conditions in the case of healthcare data processing.

As the edge clouds can be interconnected to execute offloaded tasks, these can be leveraged in conjunction to speed up the computation of offloaded tasks (particularly if applications are functionally divided so that each edge is responsible for a single application task). Resource optimization also requires to take into account different metrics, mainly latency, bandwidth, energy and cost [106]. Indeed, to optimize latency, one should also consider that network bandwidth impacts the transmission times of the workload to be executed remotely. In addition, determining the amount of processing, storage, and network resources to deploy at the edge is a difficult task, which also depends on the specific requirements on the applications that should be supported, and how they vary over time. All of these complexities have made clear that resource management is a crucial aspect to take into account, fostering, from the network management perspective, the rise of network programmability paradigms. Indeed, this is why the recent the Network Function Virtualization (NFV) and

Software Defined Network (SDN) paradigms have received increased attention in the edge cloud context, promising to make managements tasks easier and automatic. Different solutions are proposed with the aid of such technologies, for example as done in [49] leveraging SDN, while several optimization strategies are proposed and evaluated (often through simulation results) for resource management [51]. As mentioned, I have also leveraged SDN and evaluated a passive bandwidth estimation method, focusing on mobile scenarios [4].

Therefore, even with the emergence of edge cloud, it is expected that edge and core infrastructures will coexist, stressing the need not to overlook core cloud performance as new datacenters are constantly built by providers, reaching a significant coverage at the point that Mohan et al. [78] question the whole need for third party edge providers.

Finally, the aforementioned challenges in fog environments are added to the already discussed challenges of core cloud computing, for example regarding security aspects, that once again should be taken into account.

Summarizing, the adoption of the cloud paradigm and its variants has surely brought benefits for several applications, but has also opened a number of new challenges, especially concerning network monitoring, as highlighted in the next section.

1.3 Thesis contributions

I conclude this chapter presenting the main questions arising from the previous discussion, which guided the research activities presented in this thesis. As seen, the introduction and the evolution of the cloud computing paradigm has

made a whole new range of applications possible, and the evolution of cloud and mobile networks will foster innovations also in the healthcare field, where remote imaging and monitoring, just to name a few, are made possible and effective. Among the several components interacting when providing these services to final users, the network has a crucial role, and its performance, measured using different metrics, has a direct impact on the QoS perceived by customers. However, customers do not have accurate information about this aspect, as providers typically expose only qualitative information about expected performance, notwithstanding the critical importance for the applications of several metrics. Among these, the focus on latency and bandwidth in this thesis is motivated by the scarcity of works specifically targeting the cloud and mobile contexts, and the lack of real-world, in-the-wild experimentation, that, I argue, cannot be overlooked when deploying crucial services (as the telemedicine ones) onto the cloud. Therefore, to ensure that the requirements of applications running on the cloud are satisfied, there is the need to design and evaluate proper methodologies, differently according to the metric of interest.

Moreover, considering the widespread adoption of mobile devices and the additional benefits that they gain when executing cloud applications (compared to the local execution), the performance of the mobile access network has an important role as well. Even over the next few years, 4G deployment will still cover a large percentage of users, and thus its bandwidth limitations (w.r.t to 5G and wired networks) must be taken into account, as performance unpredictability is further accentuated by mobility related aspects [113, 121]. It should be also noted that early performance assessment on the currently deployed 5G infrastructures have shown small improvements compared to 4G in

terms of latency [84], highlighting that the first, radio access hop constitutes the bottleneck with around 30 ms of imposed latency.

In detail, some of the key research questions that arise are from the aforementioned challenges are the following:

- Can the current cloud infrastructure support the requirements of latency-sensitive cloud applications? How can providers and application developers accurately monitor latency and be ensured that latency experienced by users is stable and within service requirements, for example timely detecting anomalous events?

- Can mobile networks in the current state support the bandwidth requirements of demanding cloud applications? Which factors impact this metric and how can it be monitored effectively?

- Considering the complexity of cloud network environments, where new paradigms such as SDN have emerged to manage the network infrastructure efficiently, how can these be leveraged to monitor the network and satisfy application requirements?

Based on these questions, in the research activities carried on during my PhD, I have focused on monitoring latency and bandwidth on cloud and mobile infrastructure respectively, assessing to which extent the current cloud infrastructure and the mobile network can meet the requirements of multimedia and telemedicine applications, as well as others having similar requirements. In detail, the contributions presented in this thesis are manifold:

- I have measured latency from the two main public cloud providers as of today, namely AWS and Azure, during a 14 days experimental cam-

paign. Consequently, I have conducted an in-depth characterization of cloud-to-user network latency, also highlighting possible applications concerning detection and troubleshooting of abnormal situation, and evidencing the benefits of multi-cloud deployments.

- I have conducted an extensive campaign leveraging commercial mobile broadband networks (3G/4G) supported by a research testbed (the MONROE platform [7]), to assess active bandwidth estimation methods. In detail, I have assessed the use of the available bandwidth metric (at network-layer) as a proxy for the transport-layer achievable throughput, requiring a fraction of the traffic volume but still able to produce accurate and quick bandwidth estimates.

- Consequently, I have evaluated a passive state-of-the-art method for bandwidth estimation in Software Defined Network (once again leveraging the MONROE platform), assessing its accuracy in-the-wild and characterizing several factors that can possibly impact it, including traffic rate, polling period and SDN controller position with respect to the switch.

The remainder of the thesis is organized as follows. First, Chapter 2 presents the investigation of C2U network latency leveraging multiple Vantage Points and testing different Cloud Regions of the two main public cloud providers, AWS and Azure. Instead, in 3 I characterize the relationship between available bandwidth and TCP achievable throughput in mobile broadband networks, measured through active approaches. The passive approach for bandwidth estimation leveraging the capabilities of Software Defined Networks is instead the focus of Chapter 4. Finally, in the Conclusion Chapter

I provide final remarks to the activities here summarized, and I also outline opportunities for further research and investigation.

Chapter 2

Cloud-to-user network latency characterization and application

As discussed in the previous section, the last years have seen an increased adoption of services provided by public clouds, given the economical and technical benefits they provide [23].This heterogeneity of applications running in the cloud has resulted in a wide variety of Quality-of-Service (QoS) requirements, with different metric requiring fine-grained characterization within the cloud context. Consequently, a large body of literature has focused on performance analysis (for example using analytical models [10]) of cloud computing infrastructures, and on the capability of its computational resources to respond to user requests guaranteeing low response times or providing acceptable level of availability, i.e. within the Service Level Agreements.

In this context, both providers and customers showed a growing inter-

est in measurement activities targeting cloud networks, which faces several and major difficulties compared to traditional network monitoring. As stated in the introduction chapter, one of the obstacles hindering cloud adoption is that, although cloud networks are crucial to provide cloud services [65, 74], cloud providers are often unable or not willing to provide guarantees or disclose details on network performance [77]. Therefore, non-cooperative approaches [67, 90] have emerged in last years. In contrast with cooperative ones, these can integrate and expand the knowledge base that a provider is able to gather "from inside the datacenter", i.e. only leveraging a privileged view on a limited portion of the whole system. Indeed, non-cooperative approaches do not leverage privileged standpoints obtained from provider in order to obtain visibility into the main components impacting the performance of cloud-networks, namely:

- intra-datacenter networks, that is, the paths interconnecting computation and storage resources within the same datacenter,
- inter-datacenter networks, or network paths connecting the resources of the same public cloud provider located in geographically dispersed datacenter, and
- cloud-to-user networks (C2U), that is, the set of paths interconnecting users to the set of resources composing the cloud.

Among these three parts, cloud-to-user network is usually beyond direct control of both cloud providers and customers. For this reason, C2U network is harder to be monitored and (in consequence) accurately predicted compared to the intra- and the inter-datacenter networks [90, 94]. In addition, user-perceived performance is also heavily impacted by their location with respect

to the cloud resources, as the propagation delay in this case has a significant impact on the overall latency. As described, to mitigate these issues providers have typically focused on deploying more distributed datacenters in order to the network distance between users and cloud resources.

The choice of focusing on C2U networks derives from its impact on performance perceived by final users and the scarcity of available works. As discussed previously, according to the specific applications, their requirements involve the monitoring of different network metrics impacting the cloud performance (i.e. throughput or bandwidth, delay, jitter). Among these, the latency perceived by users is a critical parameter for several applications that require low latency, low latency variation (or jitter), and also both in some cases. I recall that examples of these applications include real-time video processing, cloud gaming [22] or ultra-reliable and low-latency communications services in 5G [63]. The importance of latency, which is the focus of this chapter, is also testified by the large body of literature trying to address latency issues. Briscoe et al. [18] present a detailed, comprehensive survey of literature targeting latency reduction in Internet, reviewing approaches and techniques designed in different parts of the network and at different layers of the protocol stack in order to reduce the overall latency. In detail, the work refines the latency contributions discussed in Sec. 2.1 highlighting the structural delays, the interaction between endpoints, delays related to transmission paths and to link capacities, and finally the intra-end host delays. Concerning the structural delays, as discussed in Sec. 1.2.3, edge-cloud architectures were introduced and are increasingly being deployed by providers, moving additional infrastructure and computing resources closer to the end-users to reduce the overall latency. In the context of latency monitoring, the relative novelty of

this paradigm, which is already gaining importance and is expected to grow in the near future, remarks the need to develop monitoring strategies targeted at C2U network performance. Moreover, I remark that edge-cloud architectures (as explained in Sec. 1.2.3) are often used to integrate cloud-based services rather than to replace them, and network paths towards cloud datacenter are of critical importance in this paradigm as well. In this context, several works investigate the coexistence of both paradigms: in fact, as the computational resources available in the edge cloud are limited, only tasks whose latency requirements cannot be satisfied by the core cloud alone should be offloaded to the edge cloud, in order to save edge resources. Mohan et al. [78] even evaluate to what extent the current and much more capillary deployment of public cloud infrastructures (covering more users with reduced latency) actually needs the support of edge infrastructures, and for which applications. This further motivates the need to monitor latency in the core cloud, even when edge cloud architectures will have a higher deployment compared to their current state.

For all the above reasons, this chapter focuses on the characterization of Cloud-to-user network latency, providing an extensive evaluation considering 25 geographically sparse source nodes (denoted as Vantage Points) and targeting 4 Cloud Regions for the two main public cloud providers as of today, namely AWS and Azure. The contributions in this chapter are presented as follows. Having introduced and motivated the importance of latency, I provide its formal definition and discuss the different contributions in Sec. 2.1, and then review the literature targeting latency monitoring, with focus on the cloud context, in Sec. 2.2. In Sec.2.3 I detail the experimental methodology carried on to assess C2U network latency, detailing the data collection methodology and providing background for the statistical tools leveraged in the evaluation.

The experimental results are then detailed in Sec. 2.4, considering spatial and temporal trends and comparing the two providers. Moreover, in Sec. 2.5 I provide two example evaluations to show how the latency data can be leveraged to solve real problems. In detail, in Sec. 2.5.1, I present the design and implementation of a badness event detection methodology, commenting the results obtained, while Sec. 2.5.2 discusses how measurements from different providers can be used to make informed decisions about multi-cloud deployments. Finally, concluding remarks to this chapter are given in Sec. 2.6, briefly summarizing how the results discussed can be leveraged in the context of the multimedia and telemedicine applications whose requirements are the focus of this thesis.

2.1 Latency of Internet paths

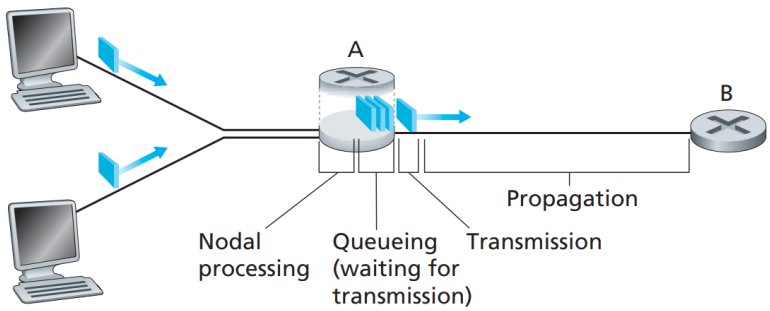


Figure 2.1: Contributions to packet delay (source [64]).

In general, a packet traversing an Internet path is subjected to different components to the overall delay [64], as reported in Fig. 2.1. In detail, the **node processing**, the **queuing**, **transmission delay**, and finally the **propagation**

along the physical medium impact on the final perceived latency.

The propagation delay is defined as the amount of time required for a packet to travel from the sender to receiver point. It therefore depends on the geographical distances between the two points and from the physical medium characteristics. For example the propagation delay in optic fiber cable can be around 0.7 times the speed of light. The queuing delay is the waiting time for a packet within a router buffer before it is processed. The transmission delay is the time required to push data onto the link, measured from the first bit of data to the last bit. As such, the transmission delay is limited by the link bandwidth. Finally, the processing delay depends of the operation performed on the intermediate nodes, and thus includes routing delays to decide where the packet should be forwarded, protocol delays (according to protocols the packet is transporting), for each link of the path traversed. In addition to the per-hop contributions, the protocol delays must be also accounted at the end hosts, considering for example the delay of transport and application layer protocols.

2.2 Latency monitoring in cloud networks

Following the increasing adoption of the cloud paradigm, seeing a huge gain in popularity over the years [1], the performance evaluation of the cloud infrastructure has attracted the interest of the scientific community, assessing the trade-offs between the multiple vantages of cloud computing (discussed in detail in Sec. 1.2) and the obstacles limiting its adoption. To this goal, some works have focused on understanding the implications of deploying specific applications

[1] Refer to previously cited Gartner report.

Table 2.1: Work dealing with latency measurements in cloud networks, including the work detailed in **this thesis**.

Net	Work	Year	Approach	Metric	Providers	Cloud multiplicity	User/VP multiplicity	Multiple probing methods	Probing period	Open Dataset
intra-DC	[96]	2018	NC	OWD	Azure, AWS, GC	2–10 VMs per CR per provider	-	N	Var.	Y
	[97]	2017	NC*	RTT	Azure, AWS, GC	6 CRs, 4VMs per CR	-	N	1 min.	N
	[95]	2017	C; NC†	OWD	AWS, GC	4 CRs, 4VMs per CR	-	Y	1 sec.	N
	[46]	2015	C	RTT	Microsoft DC	5 CRs	-	N	Var.	N
	[133]	2015	C	RTT	Microsoft DC	*2 clusters*	-	N	-	N
inter-DC	[35]	2019	NC	RTT	AWS, Azure	6–8 CRs per provider	-	N	5 min.	N
	[94]	2017	NC	RTT	AWS, Azure	4 CRs per provider	-	Y	5 min.	Y
C2U	[56]	2019	C	RTT	Azure	-	O(100M) clients	N	Var.	N
	[80]	2018	NC	RTT	Azure, AWS	4 CRs	6 VPs	Y	4 min.	Y
	[116]	2016	NC	RTT	Azure	2 CRs	5 VPs	Y	3–4 min.	Y
	[118]	2016	NC	RTT	Azure, AWS	4 CRs	6 VPs	Y	4 min.	Y
	[66]	2016	NC	RTT	10 service provs.	10 hosts overall	2 VPs	N	-	N
	[79]	2015	NC	RTT	Azure	4 CRs	6 VPs	Y	3 min.	Y
	[13]	2013	C‡	RT	AWS	-	1 VP	N	Var.	N
	[22]	2012	NC	RTT	AWS	3 CRs (US only)	≈2.5k US users	N	30 min.	N
	this thesis	**2021**	**NC**	**RTT**	**AWS, Azure**	**4 CRs**	**25 VPs**	**Y**	**1 min.**	**Y**

Legend:

*: Requires access to Time Stamp Counter register, not always available;

†: NC adoption results in higher variability due to virtualization layers;

‡: Passive analyses, traffic captured at the PoP.

Net: intra-DC (intra-datacenter), inter-DC (inter-datacenter); C2U (cloud-to-user);

Approach: NC (non-cooperative), C (cooperative);

Metric: RTT (round-trip time), OWD (one-way delay); RT (response time).

onto the cloud [117], while others have explored specific aspects of the cloud ecosystem related to cloud networks [77]. Specifically, they take into account their cost and the resulting performance and impact on user applications, with a specific focus on latency-sensitive ones [44].

To briefly recall, the limited visibility into cloud networks derives from the lack of monitoring into the network infrastructure itself and from the fact that cloud providers generally do not disclose the proprietary performance-

monitoring information about the state of their infrastructures. This has led to the design of non-cooperative methodologies in order to investigate the performance of cloud networks. These methodologies often take advantage of active monitoring approaches, in contrast with cooperative ones, which use privileged information and insider views only available to service providers or traffic carriers. Moreover, as the cloud infrastructure is leveraged by diverse applications with different goals and requirements, different portions of the cloud network can have an impact on the user-perceived performance. Indeed, different portions of the network can be identified according to paths connecting cloud resources to (i) resources within the same datacenter, (ii) resources in geographically distributed datacenters, and (iii) cloud users. These paths are denoted as *intra-datacenter*, *inter-datacenter*, and *cloud-to-user* network, respectively.

In the following subsections, I focus on monitoring and benchmarking of cloud-network infrastructures, specifically taking into account studies leveraging non-cooperative approaches to evaluate the latency experienced by end-users when connecting to public clouds. Moreover, this analysis is divided considering intra-datacenter (in Sec. 2.2.1) and inter-DC and C2U networks (in Sec. 2.2.2) separately.

2.2.1 Monitoring latency in intra-datacenter networks (intra-DC)

Increases in network latency, packet loss, or reduction in bandwidth in intra-DC networks, even in limited amount, reduce the Quality of Experience perceived by final users, thus affecting both the user's cost and the service provider's revenues [97]. Due to to their scale, the traffic volume they need to handle, and diversity of faults, these networks present unique characteristics

and require considerable effort to debug and troubleshoot, with proper tools needed to monitor different metrics with considerable accuracy and fine granularity. Therefore, several efforts have been put into allowing the provider to evaluate traffic patterns, packet drops, load imbalance [133], and especially latency [46]. Non-cooperative approaches have been also evaluated, again considering a wide range of metrics, including perceived network throughput [67, 90, 91], available bandwidth [47], and latency. Focusing on latency, ptpmesh ([95]) has been purposely designed to continuously measure the network latency (in terms of one-way delay) and packet loss inside datacenters. Leveraging the outcomes of this research, a characterization of the provider intra-DC networks for different providers has been also provided [96].

In general, measuring intra-DC performance faces several and specific challenges compared to other network environments. Indeed, computer and network virtualization, leveraged in response to scale and efficiency concerns of the providers [19], also impact on the results provided by monitoring tools [47]. The virtualization layers indeed introduce non-negligible delays and variability in the resulting performance, which are even emphasized in the case of sub-ms latency and when proper hardware configuration is not made available by the providers [96]. Moreover, different kinds of intra-DC paths may exist, leading to severe performance discrepancies [90]. Unfortunately, network topology information is usually kept confidential, even if it could be used to improve monitoring and benchmarking [99]. Finally, the impact of the management strategies implemented by providers should also be taken into account to understand performance variability [91].

2.2.2 Monitoring latency in cloud WANs (inter-DC and C2U)

According to research trends and latest reports [24], the interest in monitoring the performance and the QoS of the cloud wide-area networks is growing, for what concerns both inter-DC and C2U networks. Indeed, providers have made huge investments in specific technologies and cutting-edge solutions with the goal of improving availability, manageability efficiency, and performance, for example by deploying proprietary WANs [94], novel CDN solutions [92], or complex overlay services [21].

In this context, several works focused on investigating the performance of cloud WANs, considering throughput [35, 58, 92, 94], availability [132], latency [35, 44, 58, 94], etc. In some cases, unexpected results were reported, for example resulting from the fact that inter-datacenter connections do not always benefit from proprietary links [94].

Concerning C2U latency, according to the surveyed literature, limited attention was given to latency monitoring and analysis leveraging cooperative approaches [13, 56]. In detail, Jin et al. [56] take advantage of data coming directly from the Azure provider, consisting in Round Trip Times (RTTs) derived from Transmission Control Protocol (TCP) handshakes. Using on this information, active proving can be performed selectively, saving monitoring resources, in order to locate issues and faults more precisely. Conversely, Bermudez et al. [13] explore AWS traffic characteristics and response time through a passive analysis performed from a privileged vantage point deployed at the Point of Presence (PoP).

As data at this refined granularity is not publicly available, most of the works focusing on C2U latency either exploit datasets and information not de-

rived from cloud measurement or collect data via *active probing*. For example, some works apply measurement-oriented approaches to evaluate the deployment of hypothetical cloud services in different geographical locations [122], estimating the number of cloud infrastructures to deploy to meet application requirements in terms of latency and throughput. Others analyze the currently deployed datacenters, observing that it is sufficient to provide users around the globe with the necessary quality of experience in terms of response times (20–200ms) for interactive, latency-sensitive applications [44]. Instead, Choy et al. [22] evaluate latency from 3 AWS datacenters towards thousands of users (selected among active BitTorrent clients), focusing on endpoints located in the US, to assess the feasibility of the cloud gaming application, as one of the most demanding use cases requiring real-time interaction and high-quality video streaming. As outcome of this analysis, authors highlight the need to move the infrastructure towards the edge in order to satisfy the stringent requirements of the considered scenario, also stressing on the importance of real cloud measurements to investigate the characteristics of current infrastructures. Laghari et al. [66] evaluate RTT values towards endpoints involving ten different cloud and service providers (including Salesforce, Facebook, etc.) from two VPs (located in China and Pakistan). Their experimentation also tests real 3G/ 4G broadband networks to collect latency measures. However, the results analysis in this work only provides a simplistic characterization, reporting the average response time and the Mean Opinion Score for the video platform testing. Contributing to the effort of providing a public dataset for cloud latency measurements, Tomanek et al. [116] present a platform for collecting latency measurements from distributed source nodes, or Vantage Points (VPs), named CLAudit (acronym for Cloud Latency Auditing platform), choosing Azure as

provider. In detail, latency measured refers to C2U RTT, and is collected at different TCP/IP-stack layers, adopting different probing methods. Developing on the collected data, the same authors present a detection methodology for suspicious events [79]. This methodology considers the different contributions to latency, and defines different metrics in order to pinpoint the cause of anomalies and perform troubleshooting according to an event tree. In later works, CLAudit was expanded to include additional measurements involving the AWS provider; these data are then leveraged by Mulinka et al. [80] to detect anomalies via unsupervised learning, comparing several clustering approaches and demonstrating an accuracy improvement over the metric-based detection method previously presented in [79]. These additional data are also leveraged by Uhlir et al. [118] to evaluate a benchmarking methodology to compare cloud providers. This latter work is however focused on simple user-defined metrics such as mean latency, standard deviation and coefficient of variation. Moreover, this work does not provide an in-depth evaluation of the methodology, but provides an example evaluation considering a restricted scenario. Finally, the results from multiple source points are aggregated, therefore not investigating outcomes related to specific VPs or geographical regions.

Moreover, more recent works (conducted after the experimental work presented in this thesis) have already proposed additional contributions. In detail, Mohan et al. [78] explicitly address latency issues in current cloud infrastructures, which have reached a considerable deployment, experimentally evaluating which applications effectively require edge deployments. While the experimental campaign conducted is massive, with 101 tested CRs from 7 providers, and more than 3200 VPs tested with the aid of the RIPE atlas platform, covering different access technologies, the measurements (made publicly avail-

able) are only conducted using the Ping tool and are collected every 3 hours. Moreover, the overall characterization only focuses on the minimum latency experienced by VPs in different continents, with the main goal of assessing the feasibility of latency-sensitive applications using the current public cloud deployments rather than providing a comprehensive C2U latency characterization.

The results of the above analysis are summarized in Table 2.1, where I highlight that I have reported the most relevant literature published prior to the experimental work presented in this thesis and that has therefore influenced the design of the experimental campaign described herein. For this reason, I have not included in this table the work presented by Mohan et al. [78], which was discussed in detail above.

From this analysis, it appears evident that most of the literature focusing on C2U latency monitoring through non-cooperative approaches is based on the data collected via CLAudit platform. However, each work focuses on a specific subset of the whole data, for example considering different providers, number of probe types, period between each measurement, number of VPs and Cloud Regions (CRs). Compared to the works analyzing C2U latency via active probing, the work detailed in this thesis considers the same number of CRs and both AWS/Azure providers, but employs a higher number of VPs, i.e. 25 VPs as opposed to only 6 deployed by CLAudit, covering a larger geographical area. I highlight that number of nodes in the presented campaign would be higher even if counting the secondary and backup nodes deployed in the CLAudit platform, reaching a total of 15 VPs. Moreover, in addition to probing methods already included in previous analyses [79, 80, 116, 118], I also consider HyperText Transfer Protocol (HTTP) and TCP measurements over

non-standard ports, therefore investigating possible policies enforced basing on the transport-layer port. Finally, I measure latency with a finer granularity ($1\,\mathrm{min.}$) w.r.t. the aforementioned works.

2.3 Experimental methodology

This section focuses on the experimental methodology leveraged to characterize C2U latency. In detail, I first provide a description of the data collection methodology in Sec. 2.3.1, discussing the experimental parameters described earlier. Also, in Sec. 2.3.2 I also provide background to two statistical tests that were leveraged in the experimental analyses to provide a statistically significant comparison between providers, in terms of punctual latency values and their variability.

2.3.1 Data collection

An experimental campaign was conducted in our research group to assess C2U latency from several distributed source, for a total duration of 14 consecutive days. Sources of measurements are denoted as Vantage Points (VPs), while destination datacenters of cloud providers are indicated as Cloud Regions (CRs). AWS and Azure are considered as cloud providers, since they retain the majority of the current public cloud market share. A total of $V_s = 25$ VPs were tested, with $1\,\mathrm{min}$ period between consecutive probes. Moreover, I included $R = 4$ distinct CRs for each provider, located in four continent continents, leading to a total of 200 measured *`(VP, CR)` pairs*. I selected the following regions, where both providers have deployed their infrastructure: `Ireland` (Europe), `Virginia` (North America), `Sao Paulo` (South

Figure 2.2: Geographical distribution of VPs and CRs in the experimental campaign. Red star markers denote the 4 CRs, while the circle markers represent the VPs, colored according to the geographical region they belong to. Orange marker = EU region. Grey marker = North America region. Green marker = South America region. Blue marker = Asia Pacific region.

America), and `Singapore` (Asia-Pacific). The geographical distribution of the VPs and CRs considered for the experimental campaign is visualized in Fig. 2.2, where red star markers represent the CRs placed by each provider, while circle markers indicate the VPs [2], colored according to the geographical region they belong to, as detailed in the legend and further discussed in the experimental results.

In addition, I leverage different probing methods to measure latency at

[2] VPs were geolocated through their domain names

different layers of the protocol stack, including:

- **HTTP** probing
- **TCP** probing
- **ICMP** probing

For the first two methods, I also sent probes towards two distinct transport-layer ports, port 80 (which is commonly associated with HTTP) and the non-standard port 54321. I remark that the choice of multiple ports represents a novelty compared to the other methodologies presented before, and whose impact is experimentally evaluated in the following sections. Finally, I employ another HTTP probing method, denoted as HTTP_DB, that includes a database query to a MySQL database local to the webserver, posing an additional overhead to the overall latency. Therefore, I test a total of 6 different probing configurations; among these, however, it should be noted that ICMP probing was not suitable for Azure datacenter due to traffic-filtering policies enforced by the provider at the time of the experimental campaign. Considering the probing period and the and the duration of the campaign, each series is composed by approximately $14k$ samples. However, all the probing methods included in the campaign are subject to errors (for example, connection reset, port unreachable, timeout), which may be the result of the VP, the CR, or the network infrastructure connecting them. These error values result in invalid samples for the time-series, which are marked with the $None$ label in the dataset. In our characterization and evaluation, I properly take into account these missing values, as described later.

Focusing on the technical and implementation aspects, I leveraged the *HPing3* software tool for ICMP and TCP probing methods; *HTTPing* was in-

stead used in the case of HTTP and HTTP_DB probing. Both tools are publicly available under Linux environments.

Finally, in order to foster reproducibility [11] and research about cloud performance assessment, the dataset is publicly available at [25].

2.3.2 Statistical tests for evaluation

Before discussing the results of the experimental campaign, I provide background for two statistical tests employed with the aim of providing a statistically-sound analysis. Indeed, to compare results across providers with statistical significance, I employ the **Wilcoxon signed-rank test** [125], which is a non-parametric hypothesis test used to compare whether the mean ranks of two populations ($\{x_i\}_{i=1}^{N}$ and $\{y_i\}_{i=1}^{N}$, respectively) differ. The statistic is calculated as follows: (a) let $\bar{N} \leq N$ be the number of pairs s.t. $|y_i - x_i| \neq 0$; (b) the non-zero pairs are given a rank $\mathcal{R}_i$ according to the increasing order of $|y_i - x_i|$ (that means that smallest $|y_i - x_i|$ gets $\mathcal{R}_i = 1$); (c) pairs with the same $|y_i - x_i|$ are given the average of the ranks they span. Given this, the statistic returned by the test is computed as:

$$W_{\text{wil}} \triangleq \sum_{i=1}^{\bar{N}} [\text{sign}(y_i - x_i) \cdot \mathcal{R}_i] \tag{2.1}$$

W_{wil} is then compared to a suitable threshold (defined according to the desired p-value). In detail, for the providers' comparison discussed in the next Sections, I use W_{wil} to compare whether statistically-significant different latency values between Azure and AWS time series are observed on a given pair. When there is a statistical significant difference, the sign of the statistic allows to discriminate the best performing provider, providing lower latency.

I instead leverage the **Levene's test** [38] to assess significant difference between variability in latency experienced by the two providers. This hypothesis test allows to assess the equality of variances among K populations. To evaluate the statistic, I denote with N_i be the number of samples of ith population and let $N \triangleq \sum_{i=1}^{K} N_i$. The score of jth sample within ith group is denoted as Z_{ij}, and defined as the unsigned residual of the mean, the median or the trimmed mean, according to the chosen metric. Moreover, having defined the per-group and overall score means as $\bar{Z}_i \triangleq \frac{1}{N_i} \sum_{j=1}^{N_i} Z_{ij}$ and $\bar{Z} \triangleq \frac{1}{N} \sum_{i=1}^{K} \sum_{j=1}^{N_i} Z_{ij}$, respectively, the statistic is finally evaluated as:

$$W_{\text{lev}} \triangleq \frac{(N-K)}{K-1} \frac{\sum_{i=1}^{K} N_i \left(\bar{Z}_i - \bar{Z}\right)^2}{\sum_{i=1}^{K} \sum_{j=1}^{N_i} (Z_{ij} - \bar{Z}_i)^2} \tag{2.2}$$

The computed statistic is again compared to a threshold variable with the desired p-value to finally determine the test outcome. In the experimental results, I leverage W_{lev} to compare latency variability as experienced by AWS or Azure on a specific `(VP, CR)` pair. As for the Wilcoxon test, the choice of Levene's test is also guided by taking into account its robustness, and for this reason I have not employed other tests (as the Bartlett's test) for the considered latency variability comparison.

2.4 Experimental results

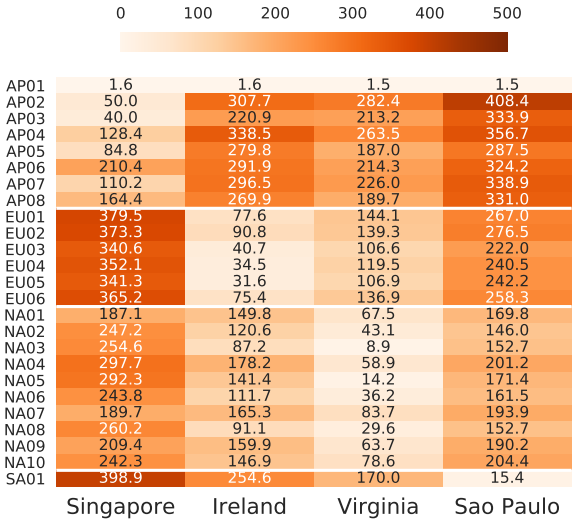
In the following, I provide an in-depth view into the experimental results obtained after the experimental campaign detailed previously. A first overall view of the results in given in Sec. 2.4.1, considering spatial and temporal trends allowed by the geographical sparse distributions of VPs and fine-grained sampling of the latency dataset collected. Also, the multiplicity of probing meth-

ods and their impact on the latency characterization is discussed in Sec. 2.4.2. Finally, the impact of human-related time patterns on latency values is analyzed in detail in Sec. 2.4.3, focusing on hourly, daily and weekly patterns observed during the 14 days campaign.

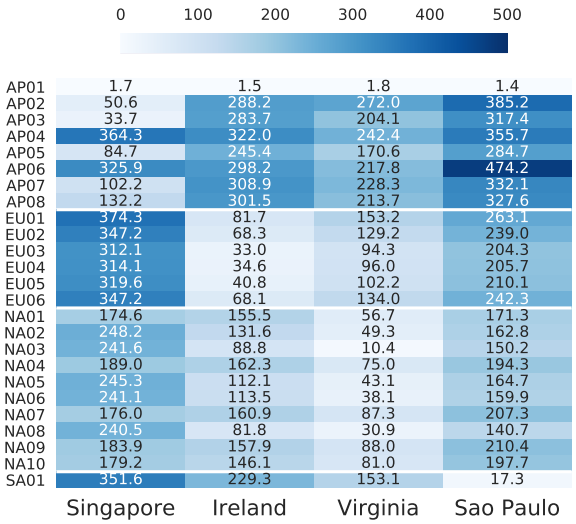
2.4.1 Characterization of spatial and temporal latency trends

In this section, I first provide a high-level view of the C2U latency results, considering each `(VP, CR)` pair separately. Complementary to this spatial analysis, I also investigate performance variability over time, thus providing two complementary views of C2U results, for a more complete characterization. Differently from previous works [116], that provided an an high-level characterization using minimum and median values, I investigate results more in detail and thoroughly. I also underline that in this characterization I do not filter out latency samples related to badness events according to the methodology introduced previously, since the goal is to provide a comprehensive view.

Accordingly, Figs. 2.3a and 2.3b report the average latency (over the 14-day campaign) experienced from each VP when targeting the four CRs for AWS and Azure, respectively, and considering TCP probing on standard port 80, delegating an analysis of the impact of probing methods to a later section. In this analysis, VPs located in the same geographical region are grouped together, according to the categorization shown in Fig. 2.2. It can be seen that latency values, as expected, grow with the distance between the VP and the CR, as lower values are reported along paths connecting VPs and datacenters within the same geographic region. The main reason for this is the propagation delay (as discussed in Sec. 2.1 when presenting the different latency contributions), that linearly increases with the distance between source and

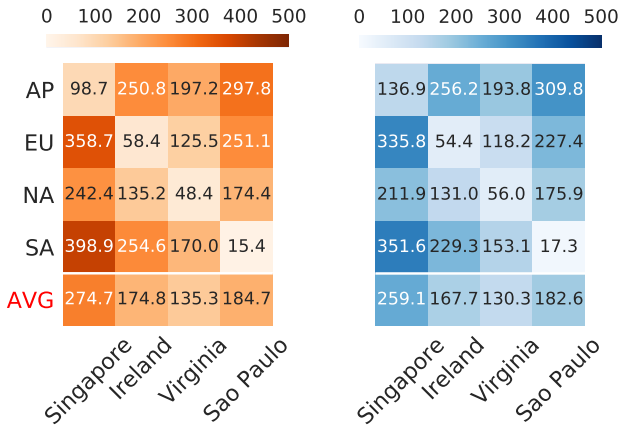


(a) AWS, detailed view.



(b) Azure, detailed view.

Figure 2.3: Average latency [ms] (14-day span, TCP probing method, port 80). (a) and (b) report detailed results considering each `(VP, CR)` pair for AWS and Azure, respectively.



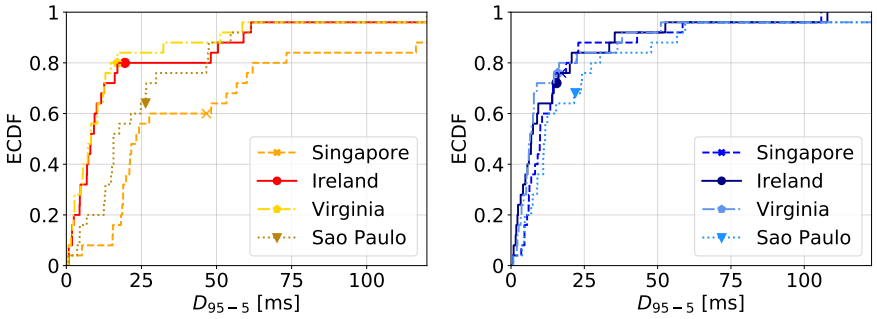
(c) AWS, per-region aggregated view.

(d) Azure, per-region aggregated view.

Figure 2.3: Average latency [ms] (14-day span, TCP probing method, port 80) (cont.) . (c) and (d) report results aggregated (average) by VP region for AWS and Azure, respectively. `AVG` reports the CR-average.

destination. However, it is interesting to note that this result does not always hold in inter-DC networks [94] or when considering other network metrics in the cloud context, such as network throughput [92], for which distance is not the main factor affecting the measurements. From these results it is also evident that VPs in the same geographical region may exhibit different behavior, thus highlighting the benefits of having fine-grained spatially distributed VPs in our experimental campaign.

In addition, Figs. 2.3c and 2.3d report an aggregated view of the previous results, obtained by averaging VP results according to the geographic zones associated to the four CRs, with an additional last row (“AVG”) reporting the CR-average. In addition to the expected lower values on the main diagonal,



(a) AWS.

(b) Azure.

Figure 2.4: Variability in terms of $D_{95-5} = 95^{th}pctl - 5^{th}pctl$. Markers highlight the average of each distribution.

On average, Virginia shows lower variability, while AWS in Singapore reports a higher (and therefore worse) variability consistently compared to the other CRs.

that correspond to latency measured within the same region, it can be seen how the `Singapore` is the CR with highest average latency for both providers, with the VPs in `SA` representing the worst case. Instead, VPs deployed in `Virginia` report the lowest latency on average for both providers. These result already provide preliminary guidelines to cloud customers wanting to deploy their applications on the cloud. Indeed supposing to leverage a single CR to reduce expenses, while considering potential users scattered around the globe, the choice of the `Virginia` region would be the optimal one in terms of average latency.

Considering a temporal analysis, Figs. 2.4a and 2.4b quantify the latency variability over time for AWS and Azure, respectively, considering to this goal

the difference between the 95^{th} and the 5^{th} percentile of the latency distribution for each (VP, CR) pair (denoted as D_{95-5}). The choice of this metric takes into account the spread between high measured values (possibly caused by congestion events, suboptimal routing, etc.) and low ones, as experienced by geographically-spread VPs when connecting to the considered CRs during the 14-day campaign. Moreover, the choice of the 95^{th} and 5^{th} percentiles (instead of $\max(\cdot)$ and $\min(\cdot)$ values, respectively) to filter-out outliers, which can be observed during high load network conditions. Both figures report the empirical CDFs (ECDF) of D_{95-5} values corresponding to the four CRs.

In detail, the figures allow to draw the following observations: *(i)* considering the per-CR breakdown, D_{95-5} is lower than 25 ms on median (resp. lower than 50 ms on average); *(ii)* the distribution of D_{95-5} shows long tails, with measured values higher than 100 ms for both providers; *(iii)* in more detail, the variability experienced by VPs in Ireland and Virginia regions is lower, on average, compared to that of Singapore and Sao Paulo, for both providers. While this discrepancy is almost negligible for Azure (e.g. $+7.5\%$ D_{95-5} for Singapore, w.r.t. Virginia, on average), this phenomenon is more evident for AWS (e.g. $+175\%$ D_{95-5} for Singapore, w.r.t. Virginia, on average). Finally, I also highlight that in most cases, variability for most (VP, CR) pairs is limited, with few notable exceptions evidencing the dependence from the CRs and the providers. These results about tail latency and variability are in line with those found by Tomanek et al. [116]. A detailed analysis has also revealed that part of the variability experienced derives from intermittent spikes that are observed for several (VP, CR) pairs, which, moreover, do not appear to follow a specific pattern. Since a decreased probing frequency reduces the possibility of observing these tran-

sient behavior, I claim that a higher sampling frequency contributes to a better characterization.

Finally, it should be noted that the proposed analysis is not designed to evaluate the trade-off between cloud costs and performance in terms of latency, since, unlike other network performance metrics (e.g. bandwidth [91, 94]) or cloud services (e.g. CDNs [92]), cloud customers are not expected to experience better latency when paying higher costs for the IaaS under evaluation. As highlighted previously, indeed, the C2U latency values are heavily dependent on the distance between VP and CR, rather than on the specific rented service (virtual machines, in this analysis).

2.4.2 Impact of probing methods

Previously, I have introduced the different active-probing methods used to measure C2U latency, covering different layers of the stack and also requiring different configurations at server side. Since these methods can lead to different latency estimates, here I assess the concordance of latency values and their variation across different probing methods. In the next Section 2.5.1, I introduce and discuss the detection of badness event, following the work conducted in [86], also evaluating the difference in detecting anomalies varying the probing method as well.

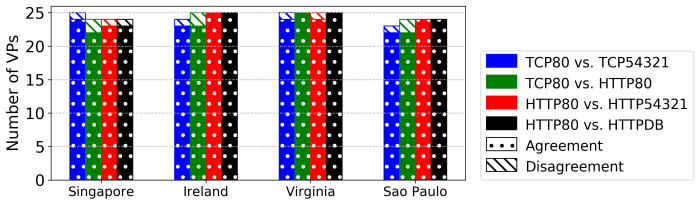
In the following, my goal is to assess whether distinct probing methods, reporting statistically different latency values when evaluated on the same `(VP, CR)` pair at the same time interval. Therefore, I leverage the Wilcoxon signed-rank test W_{wil}, used to compare the 14-day time series, leading to the following results. Considering the comparison between (a) TCP on port 80 vs. 54321, (b) HTTP 80 vs. 54321, (c) HTTP vs HTTP_DB and (d) ICMP vs

TCP 80 for any `(VP, CR)` pair and both providers (when applicable), I observed statistically-significant differences between the considered time series. It should be noted that in the third comparison (c) this discrepancy is aligned to the different operations performed by the methods, since HTTP_DB performs an additional query and therefore poses the additional latency of intra-DC network and processing time at the auxiliary server, and indeed it reported on average a $\approx 9\,\mathrm{ms}$ higher compared to standard HTTP considering all the pairs. However, in the other comparisons, there is no clear pattern emerging from this analysis in terms of which probing method performs best, although there is always one measuring a consistently-higher latency for each `(VP, CR)` pair.

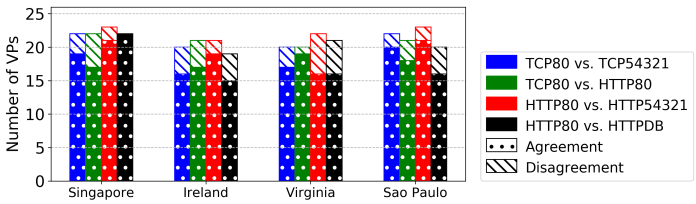
These results further motivate to need to take into account multiple methods when designing non-cooperative methodologies for monitoring public-cloud networks. Indeed, in the worst case (that is, when averaging over each time-series), these methods can differ up to 198 ms. This difference is evident in the case of the `AP01` VP, where a TCP proxy was detected on the path towards the cloud, causing measurements over the standard 80 to be heavily underestimated, since, across all VPs and CRs, the reported RTT values are around $1\,\mathrm{ms}$. Instead, RTT samples collected using the non-standard port 54321 appeared more realistic and variable according to the distance between VP and CR. Instead, for HTTP no relevant difference was reported between these two ports; this can be easily explained considering that, in this case, latency measures the response time of the requested page, which is sent from the web-server at the intended destination and cannot be sent by the intermediate proxy.

Focusing on ICMP, the results are in line with what those reported in [132] regarding service availability measurements: despite being vastly utilized, as it

does not require specific instrumentation server-side, ICMP results can differ from those obtained at upper-layer protocols, leading to either an underestimation or (more often, in this case) an overestimation of values.



(a) Number of (dis)agreements considering punctual latency. Disagreements are observed only in about 3% of the cases.



(b) Number of (dis)agreements considering standard deviation. Compared to punctual latency, more disagreements are observed (12.5% of the cases).

Figure 2.5: Number of (dis)agreements between pairs of different probing methods about best provider across different protocols in the four CRs, considering (a) punctual latency and (b) latency variability (in terms of standard deviation) as the relevant metric.

Next, I investigate the impact of the probing method in determining the best-performing provider both in terms of punctual latency and its variation, measured through standard deviation and assessing the statistical difference in these two cases leveraging the Wilcoxon signed-rank test and Levene tests respectively. The results, aggregated by CR, are shown in Fig. 2.5, reporting

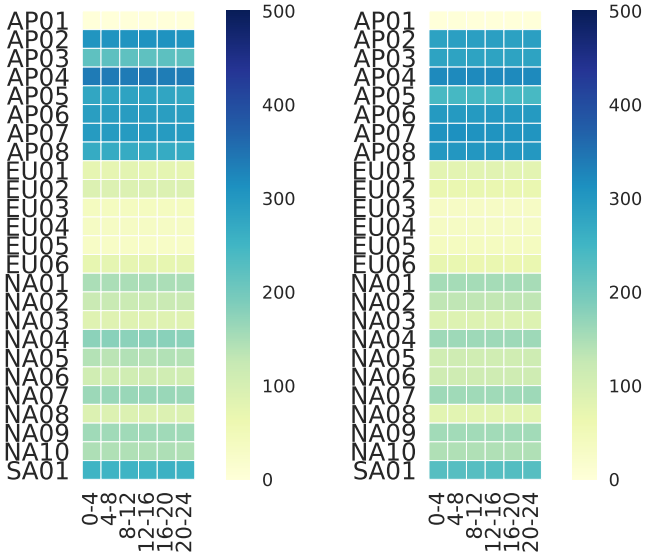
the number of agreements and disagreements between probing-method pairs considering the outcomes of the statistical tests, with Wilcoxon signed-rank test in Fig. 2.5a and Levene test in Fig. 2.5b. I remark that in some cases the bars do not sum to 25 VPs, because in these instances the test returns a non statistically-significant discrepancy between providers. From Fig. 2.5a, it can be observed that disagreements are observed only in 13 out of 400 cases (i.e. $\approx 3\%$). Instead Fig. 2.5b shows that, considering standard deviation, a higher number of disagreements is recorded (with 50 over 400 cases, or 12.5%); the highest number of disagreements concerns the comparisons between TCP80 and HTTP80, and between HTTP80 and HTTP_DB, both reporting 13 cases.

To conclude, these analyses demonstrate that, with the exception of few cases, generally the outcomes of provider comparison are not impacted by the choice of the probing method.

2.4.3 Hourly, daily and weekly patterns

In this section I report the analysis of latency over different time patterns from those considered previously. Indeed, in different contexts it is reported that traffic exhibits different patterns, reflecting human activity, so that increased traffic is observed from industrial areas during work hours and weekdays compared to nights and weekends [88]. It is therefore worth investigating if these patterns (typically referred to traffic volume) also reflect into latency experienced by users, which, as highlighted in Sec. 2.1 is impacted by link capacity (and thus spare bandwidth) and queuing delay.

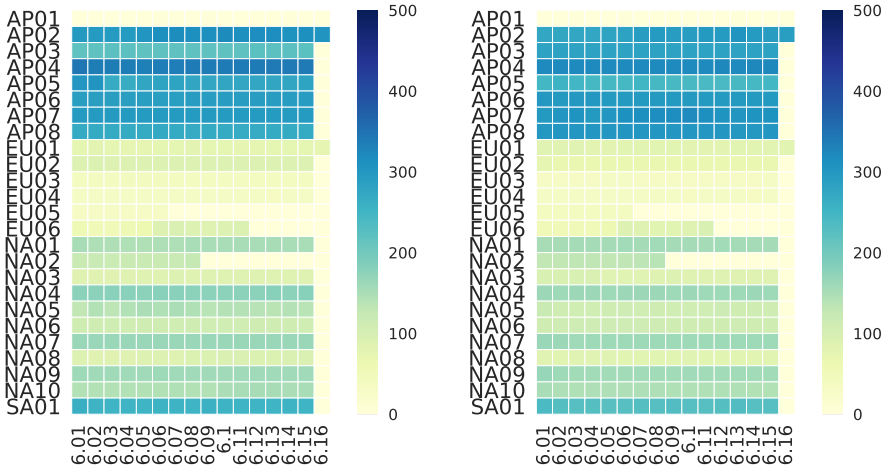
Therefore, in this section I analyze results considering three different time patterns: hourly, daily and weekly. In detail, for hourly granularity, I consider 6 time ranges of 4 hours each (starting from 12 am), and aggregate all the latency



(a) AWS. (b) Azure.

Figure 2.6: Evolution of mean latency values (in [ms]) considering hourly patterns. Figures focus on the Ireland region for both providers. Mean values are stable across different hour ranges, for all Vantage Points and for both providers.

measurements collected over the 14 days campaign according to the hour range during which they were collected. Similarly, for daily patterns I aggregate measurements for each day of the campaign. I remark that the campaign lasted for around 14 days, going from June 1st to June 16th (partially covering the first and the last day). Finally, for the assessment of weekly patterns I aggregate latency samples according to the day of the week when they were collected. As measurements were collected for approximately two weeks, this means that each weekday was covered twice during the whole campaign.



(a) AWS.

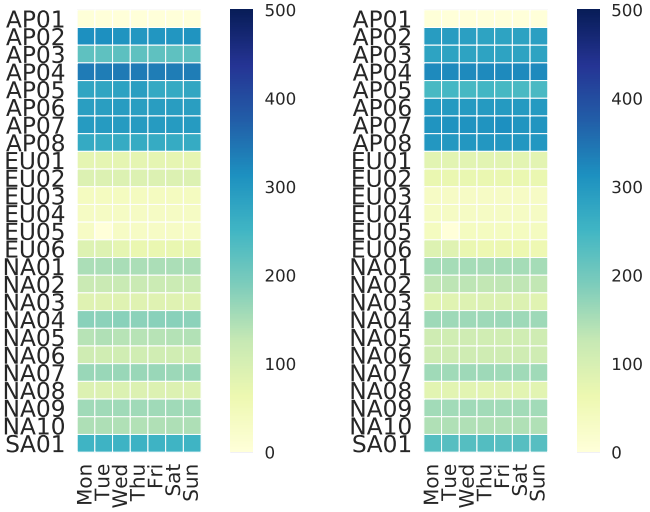
(b) Azure.

Figure 2.7: Evolution of mean latency values (in [ms]) considering daily patterns. Figures focus on the Ireland region for both providers. Mean values are usually stable across the different days, with exceptions related to `EU05` and `EU06` Vantage Points considering both providers.

Considering the results obtained after this aggregation, I report the evolution of mean latency values according to the time patterns in the heatmaps in Figs. 2.6-2.8 for hourly, daily and weekly patterns respectively. Moreover, I highlight that, among the 4 CRs, I focus on results from Ireland region for both providers, since results for the remaining CRs were analogous.

According to Fig.2.6, hourly patterns do not emerge, for both providers. Indeed, the difference between each consecutive time interval appears negligible, despite the VP or the geographic region they belong to.

While for daily patterns generally the same observations generally hold,



(a) AWS. (b) Azure.

Figure 2.8: Evolution of mean latency values (in [ms]) considering weekly patterns. Figures focus on the Ireland region for both providers. Mean values appear to be stable also across different days of the week, for all Vantage Points and both providers.

there are a few exceptions related to specific VPs. For example, latency measured by `EU05`, `EU06` and `NA02` shows a sudden drop after June 6th, 11th and 8th respectively. It is also interesting to note that 2 of these VPs are located in Europe. In these cases, the synthetic latency value averaged over the whole 14 days campaign provides a less meaningful value, as it is averaged over an almost equal amount of lower and higher values.

Finally, weekly patterns reported in Fig. 2.8 are in line with those regarding daily patterns, since there are no evident differences according to the day of the week, with the only exception of a small decrease in latency for `EU05` on

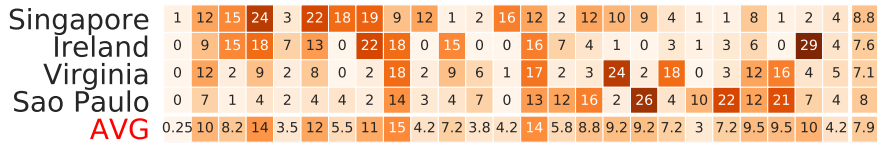
Tuesday.

Similarly, I have also considered the comparison of the two providers, using the Wilcoxon signed rank test to evaluate statistically significant difference from samples aggregated over the aforementioned patterns. The results, not shown for brevity, show with a 99% confidence that in most cases the best performing provider does not vary with the considered pattern. The only exceptions that appear are limited to the specific VPs discussed before.

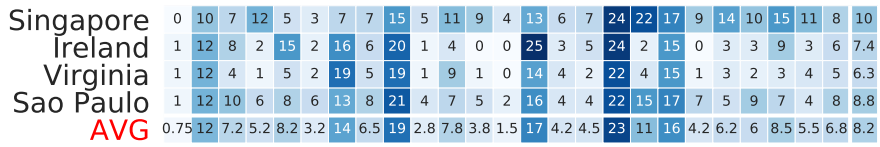
Therefore, this analysis highlights that, unlike other metrics, latency appears to be less affected by patterns related to human activity. These results are helpful in designing monitoring methodology, since providers and customers do not have to worry about monitoring the cloud infrastructure in specific time intervals, and their results can be confidently applied to other days or hour ranges without loss of generality.

2.5 Application scenarios

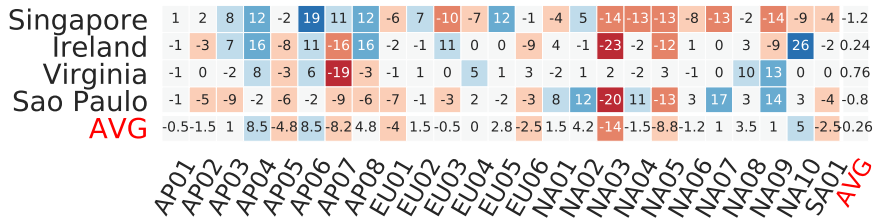
In this additional section I focus on two example evaluation of how C2U latency measured through the proposed methodology can provide concrete benefits to cloud providers and application developers, considering two application scenarios, namely the detection of anomalous latency events (in Sec. 2.5.1), and the deployment of multi-cloud applications (in Sec. 2.5.2), which leverage more than one provider, assessing the conditions where this deployment provides latency reduction.



(a) AWS, detailed.



(b) Azure, detailed.



(c) Difference, detailed.

Figure 2.9: Badness [%] in 30 min buckets (14-day span, TCP probing method, port 80). (a) and (b) report detailed results at `(VP, CR)` pair granularity for AWS and Azure, respectively. Differently, (c) reports the [%] difference heatmap. `AVG` reports either the CR- or VP-average. Average badness may differ depending on the VP or CR. However, no clear patterns emerge.

2.5.1 Badness events detection

Here, I provide an assessment of badness events for each `(VP, CR)` pair. I highlight that the methodology for badness events here described was inspired by one specific work [56], although there are different proposals [116] for

anomaly detection and interpretation. In detail, the duration of a bucket is denoted as T_{bu}, the number of corresponding samples as N_{bu}, and the vector of latency values associated to the bucket as $\mathbf{x}_{bu}$. Accordingly, each bucket is marked as *"bad"* (i.e. latency is higher than what would be expected under typical conditions) if the statistic

$$\lambda(\mathbf{x}_{bu}) > \gamma_{bu}, \tag{2.3}$$

also denoting using the summarizing function $\lambda(\cdot)$, exceeds the "badness baseline", namely a threshold separating bad and normal latency levels. For this $\lambda(\cdot)$ function, several choices are possible. In the following work, based on the characterization provided in [86], I adopt: *(i)* the median as a robust indicator of typical latency values within a bucket, and *(ii)* the 75^{th} percentile of the time series values as threshold value within each bucket, constituting a data-driven threshold to compare with the median value and detect abnormal values.

Naturally, according to the scenario a different function or threshold can be chosen, for example using provider-specified (fixed) values [56]. Both of the cited methodologies rely on the tuning of some parameters (window size, threshold for detecting events), but I have chosen the first as reference since it was deemed to be more flexible and suitable for this evaluation scenario.

Specifically, in the experiments shown hereafter, I considered two application scenarios. First, in Sec. 2.5.1, the badness threshold is computed in an offline fashion, considering the time series in its entirety, performing a post-mortem characterization of these events. Then, in Sec. 2.5.1, I consider a realistic setup by considering an online calculation which can be used to detect abnormal events in real time. In this online scenario, the badness threshold is first

initialized using the first two days of observations, for each (VP, CR), and then updating γ_{bu} when new samples are added. Moreover, samples marked as bad within the buckets are removed when learning the normal behaviour.

Finally, in both experimental analysis, a duration of $T_{bu} = 30\,\mathrm{min}$ for each bucket is chosen, with the goal of aggregating enough samples to provide statistically-significant results while keeping the duration short enough to provide the capability of timely detect anomalous events ongoing.

Offline badness events characterization

In the following, I report the results of the offline evaluation, including in Figs. 2.9a and 2.9b the percentage of bad buckets over the 14-day campaign from each VP when targeting the *four* CRs for AWS and Azure, respectively, and considering TCP probing on port 80. As in previous results, VPs located in the same geographical region are grouped together.

Overall, the average badness percentages for the two providers are similar, with 7.9% and 8.2% for AWS and Azure respectively, while results on a (VP,CR) basis show peculiar patterns. These results imply that badness events appear to be related to the VP or the destination area (or both), and generally cannot be ascribed to the provider infrastructure itself. Considering a per-VP view of the results, more occurrences of badness events are observed from EU01 and EU06 toward all the four CRs, for both providers. Differently, for NA03 (the VP reporting the highest overall average percentage toward all the CRs) and NA05 a similar behavior is reported only in the case of Azure. Instead, considering a per-CR aggregation, different behaviors can be observed according to the CR, although the relative ranking is the same observed for both providers, with Singapore, Sao Paulo, Ireland, Virginia) in

Table 2.2: Empirical conditional probability [%] of badness events.

	$X =$	
	B_{AZ}	$\neg B_{\text{AZ}}$
$Pr(B_{\text{AWS}}\|X)$	11.45	8.24
$Pr(\neg B_{\text{AWS}}\|X)$	88.55	91.76

(a) AWS.

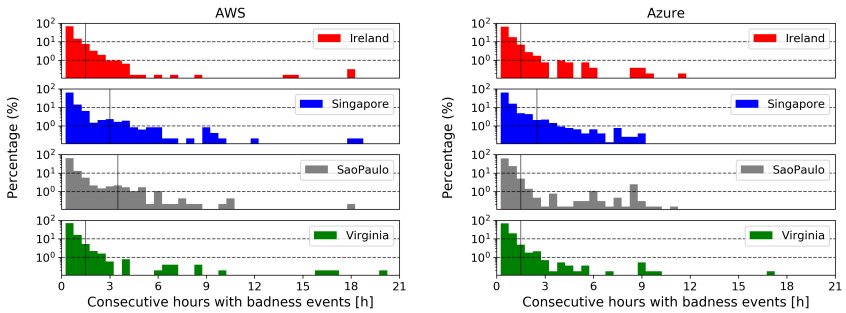
	$X =$	
	B_{AWS}	$\neg B_{\text{AWS}}$
$Pr(B_{\text{AZ}}\|X)$	11.83	8.51
$Pr(\neg B_{\text{AZ}}\|X)$	88.17	91.49

(b) Azure.

decreasing order. The highest badness percentage is experienced by `NA10` when targeting AWS `Ireland` CR; however, such result is scattered as it corresponds to neither aggregations over VPs nor over CRs.

Considering a comparison between the two providers, Fig. 2.9c reports the difference between the AWS and Azure for each `(VP, CR)` pair. The figure does not highlight specific patterns, again showing peculiarities only on selected pairs. For instance, the highest badness increase incurred by Azure (compared to AWS) is observed for `NA10` and `AP06` toward `Ireland` and `Singapore` CRs, respectively. Instead, the highest badness increase incurred by AWS (compared to Azure) is observed for `NA03` toward `Ireland` and `Sao Paulo`.

Still considering an offline characterization of badness events, I now focus on the severity of such events, assessing their correlation over time. To this goal, Figs. 2.10a and 2.10b report the empirical probability mass function (EPMF) of the badness persistence, that is, the duration of a badness event in multiples of $T_{bu} = 30\,\text{min}$ buckets, for AWS and Azure respectively. In each figure I report four EPMFs, aggregating badness persistence samples over the VPs for each CR. Results show that the distribution of the persistence of bad



(a) Badness events persistence for AWS. (b) Badness events persistence for Azure.

Figure 2.10: Persistence of badness events for both providers. The two empirical PMFs report the number of consecutive buckets reporting a badness event over the whole 14-day campaign and all VPs targeting a specific CR. Black vertical lines report the 90^{th} percentile of each distribution. For both providers, badness events are mostly short-lived (less than 1.5 hours), with few exceptions of events lasting more than 15 hours, reported especially for AWS.

buckets is similar across CRs and providers, showing an expected decreasing trend with the increasing persistence. Around 60% of the badness events lasts only for one bucket (that is, for a $30\mathrm{m}$ duration), while the 90^{th} percentile is almost always bounded within $3\,\mathrm{h}$, with the only exception represented by `Sao Paulo` region for AWS, where 10% of values are above the $4\,\mathrm{h}$ threshold approximately. These findings about short-lived anomalous events are in line with those highlighted in previous works [56, 116]. Moreover, there are some sporadic cases where badness events show high persistence (between 12 and 20h), appearing for different CRs in the case of AWS, and only for the `Virginia` region in the case of Azure.

Finally, I analyze the correlation of badness events between providers, thus considering the incidence of a badness event of one provider on the other considering the same (VP, CR) pair and the same time span. Considering a specific (VP, CR) pair, I denote with B_{AWS} and B_{AZ} the binary (i.e. $\in \{0, 1\}$) a badness event for AWS and Azure, respectively, for the same $30\,\text{min}$ bucket. I then consider the two empirical conditional probabilities, denoted as $\Pr(B_{\text{AZ}}|B_{\text{AWS}})$ and $\Pr(B_{\text{AWS}}|B_{\text{AZ}})$, assessing how much badness events from one provider reflects into anomalous behavior for the other one as well. These results are reported in Tabs. 2.3a and 2.3b, where the conditional probability is averaged over all the (VP, CR) pairs. From the tables, it appears that, when a badness event is detected for a provider, the probability that the same happens for the other one is lower than 12%. Instead, when one provider does not experience anomalous events, the probability that there is a badness event for the other one is slightly higher than 8%. Therefore, it appears that B_{AWS} and B_{AZ} are loosely correlated.

To underline the importance of the fine probing frequency adopted in this characterization, it should be noted that if a 3 times higher probing period was used, (with one sample every 3 minutes), the percentage of badness events detected would increase from 8% to 12%. Moreover, the percentage difference between coarser (3 x probing period) and finer granularity is higher than 10% for three (VP, CR) pairs, witnessing the importance of an increased sampling frequency to provide a comprehensive characterization without overestimating occurrence of anomalies.

Complementing this badness event detection analysis, I also evaluated the impact of the multiple probing methods leveraged in the campaign, assessing the agreement between different methods in detecting anomalous events. The

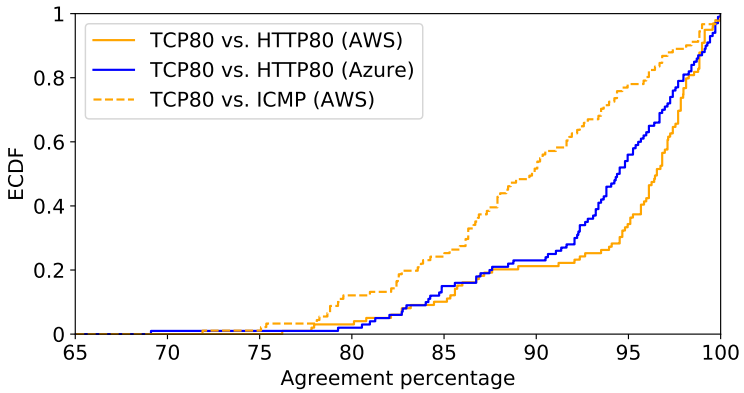


Figure 2.11: ECDF reporting the agreement percentage in detecting badness events across probing-methods pairs. Each value of the ECDF reports the agreement related to a `(VP, CR)` pair, considering the entire time series. In all cases, agreement is above 67%. Still, the median agreement between TCP and ICMP is the lowest.

results of this analysis are shown in Fig. 2.11, reporting the ECDF of percentage agreement in detecting badness events across probing methods. I highlight that the analyses take into account the agreement between TCP and HTTP (port 80) for both providers while the one between TCP and ICMP only for AWS, based on measure availability. Instead, results considering non-standard ports are omitted for brevity, since they do not differ significantly from those shown. Overall, the agreement reported settles around $(78, 97)\,\%$. In detail, higher levels of agreement are observed between TCP and HTTP (in most cases, not depending on the specific provider) with an average agreement around 93–94%. Instead, the average agreement between TCP and ICMP is around 82%, therefore reporting lower values. When looking at per (VP,CR) pair detailed results (which are not shown explicitly), it can be seen that the agreement per-

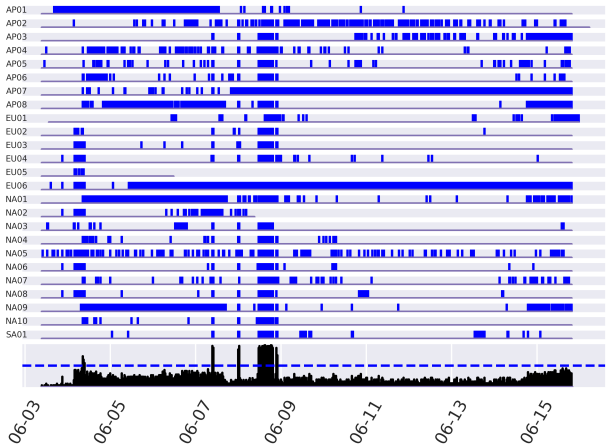
centages vary slightly with the VP but are not significantly impacted by the specific CR, with the sole exception of few cases related to TCP versus ICMP comparison.

The results about offline detection of badness events denoted the importance of the choice of the parameters for the experimental campaign, allowing to assess possible differences pertaining to VPs, CRs, providers and probing methods. Naturally, this characterization was performed post-mortem, leveraging all the knowledge acquired during the 14 days campaign. An online detection analysis is instead reported in the next section, demonstrating a possible application scenario for the proposed methodology.

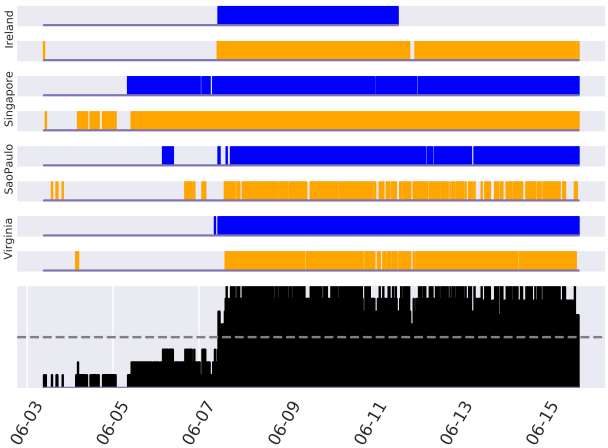
Online badness events detection

In the following, I instead focus on the online detection of badness events, that can provide a tool to detect anomalous events and means to perform a root cause analysis to detect the source of such events. Although I explicitly report the results of the proposed analysis focusing on measurements collected using TCP protocol on standard port, the same methodology can be applied and extended to multi-protocol measurements to more accurately detect the causes of degradation, and of course can be complemented by providers using measurements collected via cooperative approaches to obtain a comprehensive view.

To report the results of this analysis, Figs. 2.12a and 2.12b report a dashboard for badness events detection, where the blue and orange vertical bars report badness events (bad buckets) for Azure and AWS pairs respectively. The last row instead reports the average view, with a dashed line reporting the outcome of a "majority voting" imputation statistic (i.e. 0.5 threshold). While



(a) Online badness events (marked as blue lines) from all the VPs towards Azure datacenter in `Sao Paulo`. On July 8th, all VPs report badness events towards this CR, hinting a possible issue in the datacenter or in its proximity.



(b) Online badness events from VP `EU05` towards all the CRs for both Azure and AWS (blue and orange lines respectively). Starting from July 8th, this VP reports badness events towards all CRs and for both providers, therefore indicating a likely issue in the VP or in its proximity.

Figure 2.12: Online badness events at different aggregation. The last row reports the average of the above ones, while dashed lines indicate the 0.5 threshold for majority-voting.

this dashboard can be enriched using additional context information [56], it can provide early hints to troubleshoot cloud networks in case of performance degradation.

The two figures provide two different and complementary views of the badness events. Indeed, Fig. 2.12a highlights the correlation of badness events related to all the VPs, focusing on the Sao Paulo region for Azure. In this view, high latency values in the black box, which can be seen to occur between 7–9th June, indicate badness events observed by the majority of VPs, highlighting degradation events whose cause can be ascribed to the specific provider CR (for example due to overhead on the specific datacenter or on the related access network).

Instead, Fig. 2.12b reports a VP-based view, showing the badness events focusing on the `EU06` VP towards the four CRs of both providers. As can be seen, high and long-lasting values occur from 8th June in the summary view, indicating badness events involving all the paths departing from this VP, and for which, therefore, the providers cannot be blamed. In this case, the user access network is likely to be the cause of the performance issue.

These views provide an example of how the dashboard can be used to obtain a real-time view of performance degradation events of C2U network considering the different VPs and CRs. Based on these results, it can also be expected that an increased number of deployed VPs can increase the accuracy in highlighting root causes for badness events. These outcomes can be therefore used to highlight causes and to design solutions to reduce the impact of badness events on user-perceived performance, for example leveraging multi-cloud deployments as discussed in Section 2.5.2.

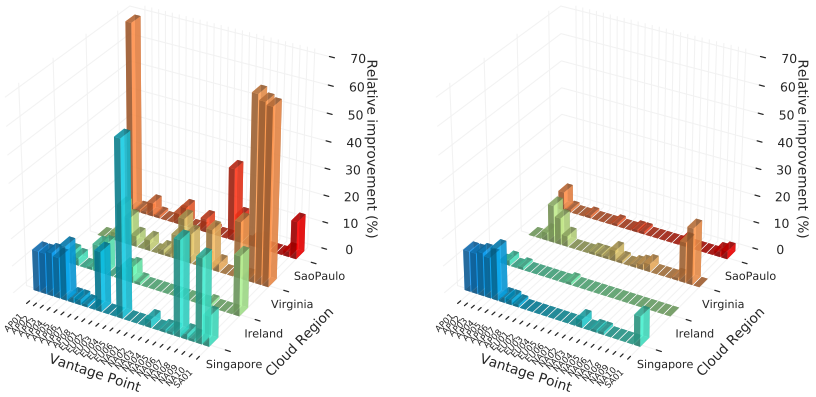
2.5.2 Multi-cloud deployments

The adoption of multi-cloud architectures by enterprises (that is, renting services from two or more cloud providers at the same time) has steadily increased over the last years, mainly because multi-cloud deployments offer more flexibility and increase reliability, also allowing to reduce costs by switching between the two providers [61].

In the following section, I evaluate the potential gains that customers could achieve when adopting multi-cloud architectures in terms of reduced network latency [127]. To this goal, I evaluate the C2U latency reduction comparing with two baseline cases: (i) the adoption of one cloud provider for all the users, considering for this case the provider reporting lower latency on average, on a global scale, and therefore Azure according to previous results. This baseline is denoted as L_{single}; (ii) the adoption of the best-performing provider considering each `(VP, CR)` pair. In this case, for each `(VP, CR)` pair, I consider to statically adopt the provider with better performance on average, based on the results already discussed. This second baseline is denoted as L_{best}).

These two baselines are compared to the performance obtained with an ideal multi-cloud architecture, considering at each instant in time the provider reporting the best performance (denoted as L_{MC}). This ideal case is representative of an architecture where either (a) the best performing provider is predicted and selected over time, or (b) resources are duplicated and redundancy is properly managed.

Figs. 2.13a and 2.13b report the results concerning the two baselines, focusing on TCP probing on the standard port 80. The former reports for each `(VP, CR)` the relative improvement with respect to L_{single}, (i.e. $\frac{L_{\text{single}} - L_{\text{MC}}}{L_{\text{single}}} \times$



(a) Relative improvement over L_{single}.

(b) Relative improvement over L_{best}.

Figure 2.13: Gains achievable with multi-cloud deployments w.r.t. the *globally-better provider* (a) and the *locally-better provider* (b). Results show that up to 70% (resp. 10%) relative improvement can be achieved over baseline in (a) (resp. baseline in (b)).

100), while the latter includes the relative improvement with respect to L_{best}, (i.e. $\frac{L_{\text{best}} - L_{\text{MC}}}{L_{\text{best}}} \times 100$). The results show how multi-cloud deployments achieve better performance compared to both cases. In detail, performance improves more than 5% in 7% of the cases, with a maximum relative improvement of 21.3% when compared to the locally-better provider. Instead, compared to the deployment relying on the globally best performing provider, latency improves more than 5% in 29% of the cases, and up to 70.8%.

2.6 Remarks and discussion

To conclude this chapter I provide some final remarks and highlight the main outcomes of the C2U latency characterization provided herein. In detail, I

have assessed C2U latency with a multidimensional dataset covering different VPs, CRs, providers, and different probing methods, collected with a fine-grained period. The analysis has shown the importance of taking into account a multiplicity of factors to properly assess latency experienced by final users, highlighting benefits for cloud providers and customers developing their applications on top of the cloud infrastructure.

Moreover, I have also focused on the application scenarios, showing how the collected measurements can provide valuable insight into cloud performance, which remains obscure although its importance for the applications is critical. Considering the requirements of telemedicine applications presented in Chapter 1, I have discussed how latency is critical for several applications. Indeed, focusing on badness events, for remote imaging transient spikes or prolonged periods of higher latency directly impact the QoS, impacting the video quality and the real-time control of devices, as in the case of telepathology and telesurgery. Similarly, the multi-cloud deployments can be leveraged to minimize latency when badness events are experienced, or leveraged in conjunction to increase reliability, another crucial requirements for telemedicine applications.

From the results shown in this chapter, it can be highlighted that the cloud infrastructures tested can effectively support these applications in several scenarios. Of course, as I am not considering application-layer delays (apart from the case of HTTP probing, which however requires a low computational efforts), the latency values reported can be considered as upper bounds to assess the feasibility of an application.

For example, considering a threshold of 100 milliseconds, that can can be considered acceptable for a human controlled application considering the typ-

ical human-interaction times [78], it can be seen that all the (VP, CR) pairs in Europe and North America report latency below this value when considering the nearest datacenter (i.e. the intra-region case). Also, the single VP leveraged in South America reports low latency towards the Sao Paulo region, since they are located nearby. Instead, only half of the VPs located in the Asia-Pacific region report a latency lower than 100 ms towards the Singapore CR (the nearest one), for both providers. In these cases, the feasibility of applications requiring real-time human interaction would be compromised. Instead, considering even stricter requirements, as a 50 ms threshold, which is typically required for interactive video streaming application (cloud gaming [98], but also telepathology in the telemedicine field), for each provider only 3 intra-region (VP, CR) pairs in Europe report latency within this threshold, while this number raises to 5 (VP, CR) pairs considering VPs in North America and the Virginia CR. These results give additional insights to providers, with a view of areas where infrastructural investments are required to enable novel applications.

In the case of mobile applications, even when the cloud infrastructure meets their requirements in terms of latency, the bandwidth limitations of these networks require additional investigations, as deepened in the next chapter.

Chapter 3

Active available bandwidth estimation in mobile networks

In this chapter I remark the importance of bandwidth-related metrics, particularly in mobile networks, considering two metrics which have received considerable attention in literature, namely available bandwidth and achievable throughput. The first is a network-layer metric independent from above protocol, characterizing the spare capacity of a path (or link), while the second is a transport-layer one, therefore dependent on the mechanisms implemented at this level. In detail, I establish a relationship between these two metrics performing an experimental campaign leveraging the European MONROE testbed, with the goal of assessing to feasibility of applications with bandwidth and latency requirements on the current mobile network infrastructure. This work complements what was done with latency, considering bandwidth as another relevant metric for several applications, and currently deployed mobile networks as a more challenging scenario compared to traditional home

networks, since both wired or wireless (i.e. WiFi) technologies can offer today a significant amount of bandwidth.

Indeed, Mobile Broadband Networks (MBB) are spreading their coverage worldwide, also reaching high speed, satisfying the need for anywhere and anytime high bandwidth access to the Internet [14]. With the increased capabilities of the cellular networks, new kind of services are made available to users, and new usage patterns emerge. Moreover, Wi-Fi services are widespread and multi-homed devices are common nowadays, offering the possibility of using WiFi together with one or even two SIMs to leverage cellular networks. In some cases, multi-carrier access is also available with a single SIM card [69]. The support of multiple heterogeneous network access providers is also specified in 5G goals [9].

In the case of multiple network paths to choose from, with possibly significant differences in performance, key factors include latency, as discussed, and throughput of each path, with variable priority depending on the specific service. In the case of throughput, one way to assess this metric involves performing a data transfer of a suitable duration, and derive the average goodput, i.e. the number of application-layer bits per unit of time, excluding the protocol overhead. Although simple, this method has different drawbacks: first, the choice of transport-layer (or application-layer) protocol, the transfer duration, which have been shown to impact the results [55]. Moreover, these kinds of measurements are costly in terms of generated traffic volume (especially when speed can exceed tens of Mbps), a decisive factor for mobile networks where there is usually a data cap.

For these reasons, the choice of available bandwidth as a network layer metric can be more suitable. This metric characterizes a path considering its

spare capacity, and is less volume intrusive compared to the previous methods; moreover, it is independent from above layers, and therefore can be considered an ideal choice to characterize the network path. However, from the surveyed literature it is evident that most of the available bandwidth estimation tools are designed for wired networks scenarios, and only few tools consider wireless and mobile scenarios, that present specific challenges given their characteristics at the physical and medium-access control layers. For example, the capacity of wireless links can vary quickly according to the radio channel conditions. For these reason, the estimation of available bandwidth in mobile broadband networks is of high interest and presents several challenges which are still a research topic.

As briefly introduced earlier, in the work presented in this chapter, I assess two alternative metrics for bandwidth, namely available bandwidth and TCP achievable throughput, evaluating the use of the first (a network layer metric) as a proxy for the second (a transport layer, protocol specific metric), bringing a considerable saving in the generated traffic volume, which is a relevant aspect in mobile scenarios. To this goal, I assessed the performance of a recent publicly available estimation tool, named Yaz [109]. I remark that, since this tool was designed for wired networks, I implemented some changes in order to take into account packet reordering, which I found to be common in my experiments using mobile networks, thus designing a novel version of the tool targeting mobile scenarios. The modified tool is also made publicly available [76]. I conducted an extensive experimental evaluation leveraging a real 3G/4G testbed deployed in different European countries, the MONROE platform [115]. Leveraging nodes from this platform, I tested actual providers using commercial data plans, in the wild, running measurements in the uplink

direction towards a measurement server hosted in our laboratory in Naples. The available bandwidth estimates obtained in such way are compared to TCP throughput measurements performed in the same conditions, highlighting the relationship between these two different metrics, and also taking into account the intrusiveness in terms of generated traffic volume and time to generate these estimates. I then focus on the conditions in which available bandwidth can be used as a proxy for achievable throughput, being accurate while keeping the amount of traffic required significantly lower. I also focus on the possible enforcement of policies by mobile network operators and how they impact the difference between the two estimated metrics. I underline that, in this sense, there are no previous works that have considered the relationship between these two metrics and how available bandwidth can be leveraged to support bandwidth estimation at a lower cost in terms of traffic volume. Finally, I remark that the dataset obtained considering both estimates (in terms of log files) is made publicly available [28], and that also the experiment code (as a Docker container image) is available [29], in addition to the modified version of Yaz discussed earlier.

The rest of the chapter is structured as follows. In Sec. 3.1, I first provide background with the definitions of the bandwidth related metrics considered in this chapter, and introduce the MONROE testbed leveraged during the experimental activities. Then, in Sec. 3.2 I survey the literature about active available bandwidth estimation tools. In Sec. 3.3 I detail the methodology used to perform the measurements on the MONROE platform, and in Sec. 3.4 I discuss the results in detail, assessing the relationship between achievable throughput and available bandwidth in mobile networks. Finally, Sec. 3.5 concludes this first chapter dedicated to available bandwidth, highlighting the connection with

latency and bandwidth sensitive applications which are the core of this thesis.

3.1 Background

Throughput and Available Bandwidth A number of different metrics exist for bandwidth, for example related to the layer of the stack they refer to. In general, network throughput refers to the amount of information transferred from a source to a destination over a time interval. Considering the TCP/IP protocol stack, the application-layer throughput is limited upper-bounded by the network-layer throughput, and also depends on the transport protocol adopted. In the work presented in this thesis, I have focused my experiments on TCP achievable throughput, although, as discussed later, I have surveyed works using both UDP and TCP throughput. I remark that instead available bandwidth, as mentioned, is instead a network-layer metric (and therefore independent from the transport protocol adopted) which measures the average unused capacity of a hop during a considered time interval, and thus can be obtained from the hop capacity subtracting the throughput flowing into that interval.

Formally, the available bandwidth is first defined on each link of a network path. For each time instant, the i-th link is either inactive or transmitting at its full capacity, so the average utilization of the link i in the time interval $(t-\tau, t)$ is

$$\bar{u_i}(t-\tau, t) \equiv \frac{1}{\tau} \int_{t-\tau}^{t} u_i(x)dx$$

where τ is the averaging timescale. The amount of traffic that is transferred over the link during the time interval $(t-\tau, t)$ is denoted as $l_i(t-\tau, t)$ and is equal to

$$l_i(t-\tau,t) = C_i \cdot \tau \cdot \bar{u_i}(t-\tau,t)$$

The available bandwidth in the time interval $(t-\tau,t)$ for the i-th link, with capacity C_i, is

$$a_i(t-\tau,t) \equiv \frac{1}{\tau}\int_{t-\tau}^{t} C_i(1-u_i(x))dx = C_i(1-\bar{u_i}(t-\tau,t)) = C_i - \frac{l_i(t-\tau,t)}{\tau}.$$

This definition is then extended to a whole path by taking the minimum available bandwidth of all the links composing the path, and can be considered as the maximum network-layer throughput that can be imposed on that path without affecting the other flows sharing part of it. The knowledge of available bandwidth allows to impose traffic on a path without causing congestion, thus resulting in bounded delays (in metaphor, it flows effortlessly as in an unloaded network). All applications sensitive to jitter (e.g. interactive applications, media streaming) or to throughput (e.g. bulk transfer) would benefit from the knowledge of the available bandwidth, for a number of uses, including for example server selection, overlay network routing, buffer dimensioning, media codec selection.

Both throughput and available bandwidth can be estimated both through *active* and *passive* measurement approaches. For available bandwidth estimation using active approaches, several tools have been proposed in literature, which can be broadly categorized into two approaches, that I briefly discuss. The Probe Gap Model (PGM) [52] approach uses probe packet pairs or packet trains to determine the available bandwidth. It uses these pairs by noting the difference between their network entry time gap, and their network exit time gap. The difference between these two gaps is the time the bottleneck link

required to service any non probing traffic (i.e. cross-traffic) on the bottleneck hop. Therefore, the information about cross traffic rate can be used to estimate the available bandwidth [111].

In the second approach, namely Probe Rate Model [41], a train of packets is sent leveraging the concept of self-induced congestion [129] to determine the available bandwidth of the network path. Each packet train is forwarded through the network at a particular rate. The rate increases until particular self-induced characteristics are observed from the packet train such as a diversion from the initial packet train transmission rate. Each of these approaches has its advantages and drawbacks, and none of them is better than the others in all the possible application scenarios.

In this chapter, I focus on active approaches based on the Probe Gab model, leveraging tools which send trains of packets, organized as *fleets*, into the network and measuring their gaps to determine network congestion and thus derive the available bandwidth. Instead, I refer to the next chapter 4 for the discussion and the evaluation of a passive methodology.

MONROE platform [7] As mentioned in the introduction to this chapter, I leveraged a research testbed for the experimental evaluation carried on in this chapter and in Chapter 4 for bandwidth estimation on mobile broadband networks. This platform, named MONROE, was developed within the scope of an European project whose objective is to provide an open platform for independent, large-scale monitoring and assessment of performance of MBB networks in heterogeneous environments. While a deep discussion of the platform can be found in [7], its main components are the following: (i) distributed standardized hardware appliances (*MONROE nodes*, also referred to as *mobile*

nodes in the following) running the experiment software; (ii) the software—-core components and user-defined experiments—running on MONROE nodes in virtualized environments; (iii) the management system, allowing users to access, schedule experiments, and import data; (iv) a database holding experimental results.

Notably to the experiments performed in this chapter are the virtualized environments, in the form of Docker containers, used to run the experiments, and the monitoring system integrated into the platform, which periodically collects metadata within the experiments, for example relative to mobile network condition (signal strength, frequency used, etc...).

3.2 Reviewing bandwidth estimation using active approaches

Several available bandwidth estimation tools have been developed over the years, mainly targeting wired networks, providing an assessment of their performances in different scenarios. The list of the most popular and used tools include, for example, Pathload [55], Yaz [109] (a calibrated version of Pathload aiming at improving the accuracy), pathChirp [101], ASSOLO [41], SPRUCE [112], Traceband [45]. These tools were introduced and evaluated mostly focusing on wired scenarios and on different conditions. For example, the authors of Traceband [45] compared its performance with that of Spruce and Pathload on a real network using different traffic patterns, showing that Traceband is faster and less intrusive than the others, and also achieving an accuracy comparable to that Pathload.

In other works, on the other hand, authors focus on providing a detailed

comparison of different state-of-the-art tools known tools, without presenting new ones, trying to provide a unified framework for the assessment of available bandwidth estimation tools. For example, Goldoni et al. [42] compared the accuracy, the intrusiveness, and the convergence time of nine of the most wide spread tools on a real testbed with 100Mbps links, and with both constant bit rate (CBR) and Poisson cross-traffic. Other works [108]-[82] evaluated such performance on very high speed networks. Authors in [16] have also shown that the combined use of different techniques can increase the estimation accuracy.

The first available bandwidth estimation tool specifically designed for WiFi networks was WBest [68], following which other more recent proposals were developed targeting wireless scenarios, including WiFi and cellular networks. For example, Farshad et al. [34] propose an enhancement of WBest (named WBest+) in WiFi networks, by evaluating the impact of 802.11n frame aggregation on probe packets and adjust bandwidth estimation accordingly. Song and Striegel [110] also leverage frame aggregation, improving accuracy by inducing it on probe packets. As baseline to assess the accuracy of their work, both consider UDP throughput, which is considered the ground truth for capacity, and the available bandwidth is estimated from it by subtracting the amount of generated cross-traffic. As such, this evaluation is only possible in fully-controlled scenarios, where the interfering traffic can be fully controlled by the experimenter. The first available bandwidth proposal designed for LTE networks is presented in [87], leveraging a curve-fitting approach to detect the turning point in the received inter-packet spacing of the probe packets train. An experimental evaluation is conducted over a Japanese commercial LTE network, and FTP throughput is used as reference for comparison. Oshiba et al.

[85] provide an assessment of the impact of LTE packet scheduler, introducing a novel tool, named PathQuick3, which also leverages a curve fitting approach to determine the transition point in a train of packets covering different rates. In this case, packet trains are designed to have a fixed duration, keeping the inter-packet time constant. Different data rates are instead obtained by changing the packet sizes. The experimental campaign is conducted on a commercial LTE network in several areas of Tokyo, and the TCP throughput as measured by speed test applications is used as reference to evaluate the error. The same authors present a novel tool in [103], named PathML, leveraging four distinct machine learning techniques, that are trained using the queuing delays measured receiver-side, with the goal of predicting the available bandwidth. The experimental campaign is conducted similarly to the previous work, using TCP throughput as reference and demonstrating an improvement over PathQuick3.

Summarizing, a first issue to be tackled for a proper available bandwidth estimation is the choice of the right tool according to the scenario. Another important issue for the available bandwidth estimation is the fact that most of the existing tools can provide accurate results only if properly used and calibrated. This issue has been revealed and analyzed by different works in literature. For example, in 2004 Paxon et al. [89] claimed that a better design stage of measurement experiments is of great importance to avoid frequent mistakes, mainly due to the imperfections of tools. The authors basically reported that a calibration is usually needed to detect and correct possible errors. Other works [8]-[111] analyzed commonly used tools on real testbeds and reported several pitfalls in which they can typically end. Furthermore, from the discussion of the literature provided above, it appears that there are several limitations in the proposals designed for mobile 3G/4G networks. First, limitations can be found

in the experimental campaigns, which are performed on commercial networks but consider restricted scenarios, focusing on only one provider and one country. Often, TCP throughput is used as reference for available bandwidth; in two out of three of the surveyed works this value is obtained through external speed-test applications, which do not provide a full insight on how this value is obtained. Therefore, there is no extended analysis of throughput, only using a synthetic (for example, the mean) value. Notably, none of the tools designed for cellular networks is made publicly available, limiting the reproducibility of the experimental results presented or their application to different scenarios. Finally, the experimental evaluations of these tools are focused on the downlink direction only, meaning that traffic is sent from a remote host/server to the mobile node, justified by the higher volume of traffic sent in this direction. This leaves an uplink analysis missing, although nowadays there are several applications requiring the transmission of data from a mobile device to the remote server, especially in the IoT and the telemedicine field. Considering indeed the rise of Internet of Things (IoT), uplink traffic is expected to continue to increase significantly, and therefore requires more attention on its performance, also considering its reduced bandwidth with respect to downlink. Therefore, I claim that such estimation is missing from the experimental literature although the mobile scenario, as discussed in more detail in previous chapters, is of practical and growing interest nowadays.

With the work conducted and summarized in the following sections, I address all of the aforementioned shortcomings as follows:

- I adopt open source tools, and publicly release the source code when making changes;

- the experimental campaign is executed leveraging an open real-world testbed (the MONROE platform);
- the effect of traffic policies enforced by operators on TCP throughput is analyzed in detail;
- different Mobile Network Operators in different countries are analyzed, highlighting the need for such diverse characterization in the results;
- I target the uplink direction, as it is more bandwidth limited direction and was neglected in previous works;
- finally, I also discuss traffic volume and time required to generate available bandwidth estimates.

3.3 Methodology and preliminary experiments

In this section, I focus on the methodology employed for the experiments, providing a description of the tools employed and the scenarios considered. I also discuss the results of some preliminary experiments that helped the design of the subsequent campaigns. In detail, I have leveraged a research testbed provided within the scope of the MONROE project in order to assess the performance of available bandwidth estimation tools in mobile scenarios, and comparing them with TCP achievable throughput measurements. Therefore, my scope is on network and transport layer measurements, according to the considered scenario, which provides partial visibility on the lower layers and does not allow changes in their configuration, similarly to most non-rooted commercial smartphones. I decided the focus on the the uplink direction given the scarcity of works targeting this direction. This means that traffic flows from

Table 3.1: Details of the experimental campaigns performed for active bandwidth monitoring.
The tools Yaz and D-ITG are adopted in all campaigns; in addition, the Preliminary one also includes Pathload.

Campaign	**Run duration (s)**	**Daily exp.**	**Campaign duration (days)**
Preliminary	60	1	8
Long-term	20	1	35
Additional	10	24	9

the mobile node towards a measurement server hosted in the research laboratory hosted in our University, connected to the Internet through a 100 Mbps Ethernet LAN and the 1Gbps backbone of the Italian Research Network Consortium (GARR) [36], pushing the bottleneck at the radio access link on the mobile node.

At the transport layer, I focus once again on TCP as protocol given it is widespread in 3G/4G networks [53], also keeping into account the lower amount of generated traffic compared to UDP given the enforcing of flow control mechanisms. Indeed, although the link capacity does not differ from the transport-layer protocol used, UDP traffic generated on the interface is accounted by the platform, even if its rate exceeds the link capacity and is not physically sent on the network.

Considering the hardware constraints of nodes employed in the MONROE project I first assessed their traffic generation capability. Leveraging D-ITG as

traffic generator [17, 31], I have therefore assessed that the nodes can saturate the nominal uplink bandwidth of 3G/4G networks, obtaining rates higher than 100 Mbps, confirming that the virtualized configuration of MONROE nodes does not hinder the experiments here described.

As additional step, I employed two available bandwidth estimation tools, namely Yaz [109] and Pathload [55], with the goal of assess the one providing available bandwidth estimation closer TCP achievable throughput measures, obtained using D-ITG set to generate traffic at a rate higher than the mobile link bandwidth. These tools are also chosen in continuity with previous works performed in the context of the MONROE project [2, 3]. In these experiments, I alternatively executed 4 consecutive runs of Yaz, Pathload (hereinafter ABW_1 and ABW_2), and D-ITG (*TAT*). This means that the order of execution of the estimates is as follow: ABW_1–*TAT*–ABW_2, followed by 5 minutes interval, then ABW_2–*TAT*–ABW_1. In this way I try to counter possible bias of the results according to the order of execution of the tools, while keeping the amount of time between the available bandwidth and the throughput measurements constant. The duration of each run during these preliminary tests was set to 60 seconds for Yaz and D-ITG, while Pathload does not employ a fixed duration, since the probing process depends on the measured conditions of the network. I ran these experiments once per day deploying nodes in the 4 countries provided by the MONROE testbed, namely Italy, Spain, Sweden, and Norway, for a total of 8 consecutive days. Moreover, the tests were scheduled over the night, in the 0–3 AM time slot (local time) in order to reduce cross traffic and variability of network conditions. With one different operator per country, for a total of 4, I underline that this is also a richer scenario that the ones considered in previous works. Finally, leveraging the platform capability, several

metadata were collected, including GPS position, node CPU usage, and modem information such as signal quality indicators (such as RSSI and RSRP), cell ID, and frequency band. These metadata were helpful to provide context for the experimental results, for example allowing to distinguish 3G from 4G connections, correlate measurements with signal strength and assess that CPU was not a bottleneck for the running tests.

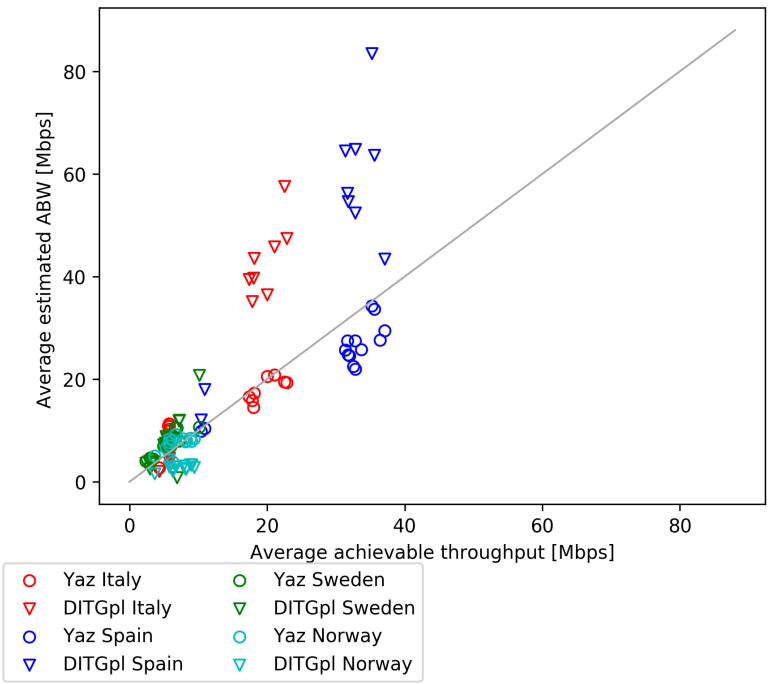


Figure 3.1: Results of the preliminary tests involving Yaz and Pathload. The difference between available bandwidth and achievable throughput depends on the country. Pathload shows higher variability compared to Yaz.

These preliminary results are shown in Figure 3.1, reporting the measured

TCP throughput and the corresponding estimated available bandwidth, each point representing the average of the 4 runs of each test. From the results, it can be observed that the results, in terms of difference between measured throughput and available bandwidth, depend on the country (and the operator in consequence). Moreover, I found that Pathload exhibits higher variability when compared to Yaz in the same conditions, and also its estimates are farther from the achieved TCP throughput. This is more evident in Italy and Spain, where Pathload estimation of available bandwidth estimated are significantly higher than the achievable throughput. The complete results also highlight that the estimates do not depend on the order of execution of the tools. Therefore, the results of this preliminary campaign were taken into account in designing the longer one, where I leveraged Yaz and D-ITG (therefore discarding Pathload) for available bandwidth and achievable throughput tests respectively, while remaining the other parameters unaltered for what concerns the number of runs and the time window. Instead, the duration of each run was reduced to 20 seconds, allowing to reduce the amount of generated traffic, but, as highlighted by the preliminary campaign, without impacting the relationship between available bandwidth and TCP throughput. Indeed, after such duration TCP throughput has already extinguished possible transitory and is stable, and Yaz can collect a sufficient number of estimates. I ran one daily experiment on the same nodes deployed in the four countries listed before, for a total of 35 days, from February 17^{th} to March 26^{th} 2018, excluding the March 8^{th}, 9^{th} and 23^{rd}, due to platform maintenance.

Table 3.1 summarizes the details of the experimental campaigns whose results are reported in this chapter. These include the preliminary one here discussed, the longer one which was just briefly introduced, and an additional

one as discussed in the following.

3.4 Experimental results

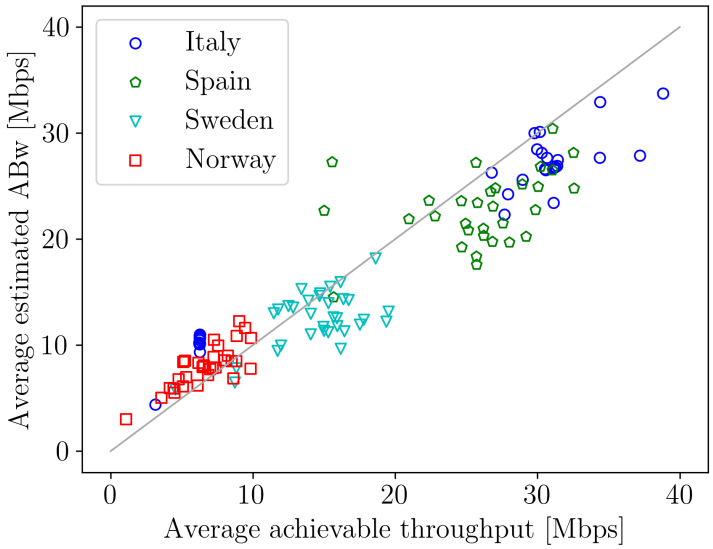
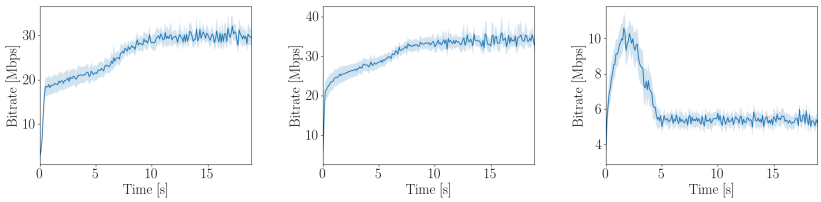


Figure 3.2: Comparison between Available Bandwidth and TCP Achievable Throughput measurements, averaging over the 4 runs. While the specific relationship between available bandwidth and achievable throughput depends on the country, it is evident that available bandwidth can be used to approximate throughput.

In the following section, I report and discuss in detail the results of the experimental campaigns introduced in the previous section, excluding the preliminary campaign, whose results were already discussed and that guided the following ones.



(a) Spain. (b) Italy using LTE. (c) Italy using 3G.

Figure 3.3: Time series for TCP achievable throughput, including the samples from all experiments. Time is calculated from the beginning of the experiment, and the shaded area represents 95^{th} percentile bootstrapped confidence interval over the different runs.

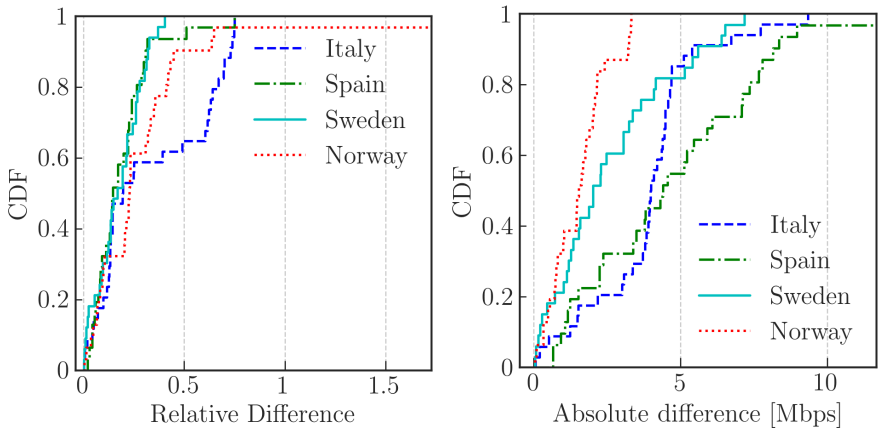
3.4.1 Available bandwidth vs. achievable throughput estimates

First, Figure 3.2 reports the results for each day of experiments, where each point represents the average of the 4 runs. The x-axis reports the achievable throughput, while the y-axis the available bandwidth, and therefore the plot allows to visualize the correlation between these two metrics as estimated by the different operators on the different nodes. I also highlight that results for three days for Spain and Norway are missing, due to traffic-cap thresholds (specific for each node and operator) enforced on the MONROE platform and to which experimenters are subjected to. Overall, it can be seen that each country reports similar values across different tests, both in terms of achievable throughput and available bandwidth. Indeed, I recall that the daily experiments are performed over the night and with the goal of minimizing variability. The only exception to this behavior, for both metrics, can be observed in Italy, as two classes of values can be observed: lower ones, grouped around 7–8 Mbps on the x-axis, and higher values, reporting around 30 Mbps measurements.

Leveraging to the metadata collected by the platform during the experiments, I underline that this behavior was caused by different access technologies employed by the node during the experiments, switching from a 4G network to a 3G one for some experiments. In these groups two distinct relationships between throughput and available bandwidth can be observed: indeed, with the lower average throughput provided by the 3G connectivity, Yaz reports overestimates, while the higher throughput obtained using LTE is underestimated by the available bandwidth provided by the tool. I underline that this behavior for larger throughput is also observed in Spain, with the tool reporting available bandwidth estimates lower than TCP throughput in almost each run.

To better understand the reasons of this behavior, I analyzed the time series of achieved TCP throughput on the receiver side, considering 100 ms intervals and focusing on Spain and Italy as the two countries where this phenomenon was observed principally. I report these time series for Spain in Figure 3.3a. In this case, it can be seen that there is a consistent behavior across runs, with throughput increasing steadily for about 10 seconds and then stabilizing. Instead, results for Italy with LTE are reported in Figure 3.3b, highlighting a similar behavior, with throughput stabilizing after a shorter interval (around 5 seconds), and with slightly lower underestimation of the achieved TCP throughput.

Instead, the behavior in Italy when a 3G connectivity was employed are shown in Figure 3.3c. This plot can provide an explanation of the underestimation provided by Yaz, since in this case throughput exhibits an opposite behavior compared to previous cases, since it first increases until a peak value, but then quickly decreases to a lower value, to which it stabilizes after around 10 seconds. This behavior was consistently observed over all runs involving



(a) (b)

Figure 3.4: CDFs reporting the difference between TCP achievable throughput and Available Bandwidth estimates ((a) Relative and (b) Absolute) in the 4 different countries.

3G networks, and I can therefore speculate it derives from a policy enforced by the network operator. As a consequence, the estimated available bandwidth is farther from the average achieved rate, but actually resembles more closely the achieved throughput during the first part of the run. Considering the traffic generated by the available bandwidth estimation tool, it consists of short-lived UDP flows, with around 50 packets per flow, with the same destination port and separated by around 500 milliseconds intervals, for a total duration equivalent to that of the achieved throughput measurements. Rather than flow duration, instead, the policy enforced by the operator may discriminate different transport-layer protocols.

In order to quantify the relationship between average TCP Achievable

Throughput (denoted as TAT) and the Available Bandwidth (denoted as ABw) estimates, I report in Figures 3.4a and 3.4b the cumulative distribution of their absolute Relative Difference (RD), which is defined as $RD = \frac{|TAT-ABw|}{TAT}$ and also considering the Absolute Difference, which is not normalized on the average throughput.

From the figures, it can be seen that Yaz estimates considering Sweden match more closely the throughput in terms of relative difference. In Norway, instead, in some cases the achieved throughput is low (lying around 1 Mbps), while the estimated bandwidth is around 2 Mbps, leading to a high, (almost 100%) relative difference. This explains why the absolute differences reported for Norway are the lowest, but at the same time since the throughput is also lower, the impact on the relative difference is significant. In this specific case, however, the high relative error would not heavily affect applications that only have bandwidth requirements in terms of a minimum threshold. Finally, I underline that the absolute difference is higher in Spain, and this can be also explained in terms of the throughput behavior reported earlier, exhibiting a longer transient.

3.4.2 Generated traffic volume

Given the importance of traffic caps in Mobile Broadband Networks (where exceeding traffic is expensive), I aim to quantify the benefits introduced by using available bandwidth tools, comparing their intrusiveness in terms of generated traffic volume to that of tools measuring the achievable throughput.

I compare the traffic volume generate to estimate available bandwidth and for achievable throughput estimation in Figure 3.5. In this plot, the diagonal line represents equality; therefore, it can be seen that Yaz estimates always

Table 3.2: Average traffic volume generated over 4 runs lasting 20 seconds each.

Country	Achievable Throughput (MB)	Available Bandwidth (MB)	Ratio
Italy	210.38	18.61	11.30
Norway	72.83	8.54	8.52
Spain	263.03	17.94	14.65
Sweden	149.57	15.54	9.62

require considerably less traffic to produce an estimate. Moreover, different countries and operators can be grouped differently, and also the 3G/4G connectivity in the case of Italy can be distinguished, according to what was discussed earlier. In detail, available bandwidth and achieved throughput volume under 20 and 100 MB respectively are related to experiments employing 3G connectivity.

Interestingly, it can be seen that as the network performance increases in terms of achievable throughput, the traffic volume required by D-ITG for these tests grows significantly faster than that generated by Yaz to estimate available bandwidth. Also, traffic generated for estimating the available bandwidth ranges in a smaller interval (3–26 Mbps) and are therefore more stable and predictable compared to the volume generated by achievable throughput measurements (ranging between 19–393 Mbps). I summarize these results in Table 3.2 reporting the average volumes of traffic generated for available bandwidth and

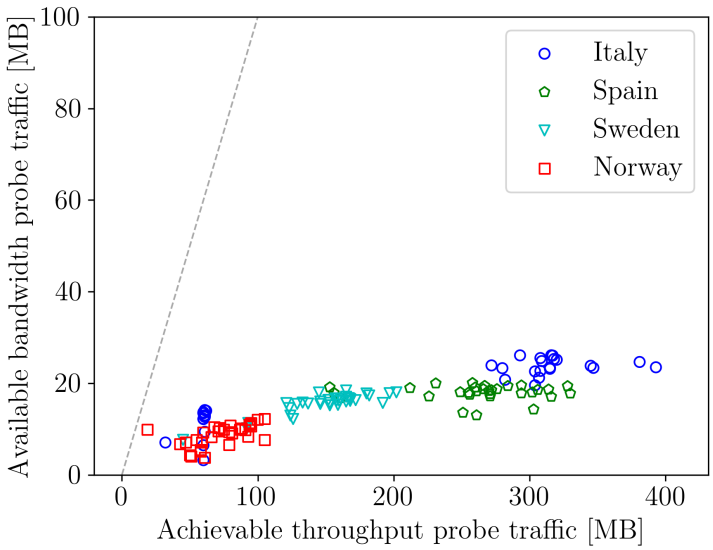


Figure 3.5: Comparison between traffic volume generated for Available Bandwidth and Achievable Throughput estimations. Dashed line represents equality.

achievable throughput, and also their ratio to ease the comparison. As can be seen, in Spain available bandwidth measurement require almost $15\times$ less traffic compared to achievable throughput tests. This result further highlights the benefits of estimating available bandwidth and using it to infer achievable throughput, gaining increasing importance in future high-speed mobile networks, as for 5G networks targeting speeds over 1 Gbps.

3.4.3 Measurement latency

I recall that in order to obtain each available bandwidth estimation, a variable number of streams is sent, terminating this iterative process when the difference between local and remote packet spacing is below a (configurable) thresh-

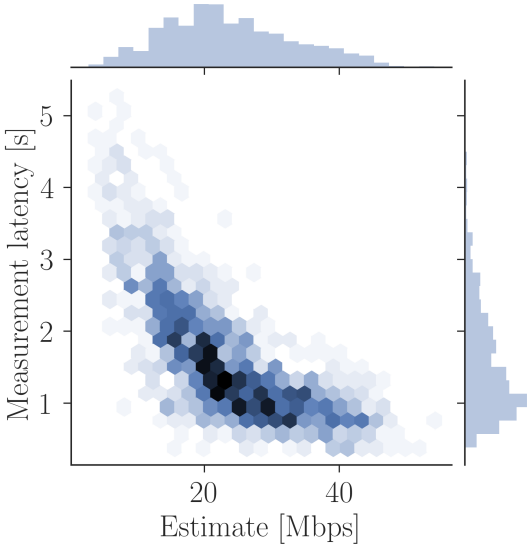


Figure 3.6: Joint density distribution considering results versus measurements latency for available bandwidth estimation in Spain. Marginal distributions are also shown on each axis.

old. For this reason, the network condition encountered by the probing streams impact the duration of the estimation process, i.e. the measurement latency. I highlight that during the experiments it was observed that this latency depends on the number of streams generated, according to a linear relationship. No abnormal behavior has been observed, for example related to the TCP connection establishment or termination, used by the tool for the signaling channel. Measurement latency impacts on the responsiveness of the estimation tool, but also on the traffic volume generated, and therefore is a metric worth investigating.

Figure 3.6 reports the joint distribution of measurement latency and estimated values for ABw as obtained in our 35 days campaign, focusing on Spain as country for the peculiar patterns observed therein. Indeed, as can be see, in

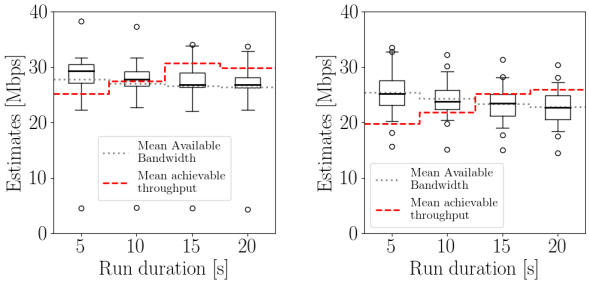
Table 3.3: Mean and standard deviation for measurement latency, for each of the 4 countries tested.

Country	Mean (s)	Std (s)
Italy (LTE)	1.15	0.52
Italy (3G)	0.79	0.32
Norway	1.32	0.79
Spain	1.60	0.83
Sweden	1.41	0.80

this case the time to generate an estimate decreases for larger estimated ABw values. Explanations for this behavior partially include the algorithm implemented by the estimation tool, since the first streams are sent at a higher rate, and thus more streams are needed to converge to a lower value of bandwidth; in addition, this trend is also related to the available bandwidth variability over time, which is higher in the case of poor network conditions.

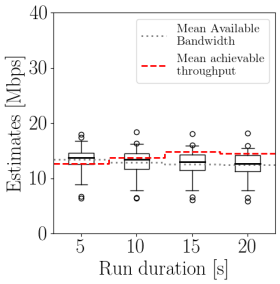
In the other countries, the estimates report lower variability with some exceptions relative to few cases requiring more than 3–4 seconds when the estimated value is low. Quantitative results about measurement latency for each country are reported in Table 3.3. Overall, it can be seen that on average the time required to generate an estimate with LTE ranges from 1.1 to 1.6 seconds depending on the country, but the variability in terms of standard deviation is not negligible, with up to 0.83 seconds of deviation from the mean.

3.4.4 Measurement latency vs. accuracy

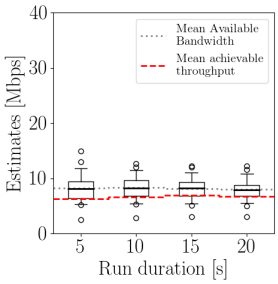


(a) Italy.

(b) Spain.



(c) Sweden.



(d) Norway.

Figure 3.7: Available bandwidth estimates boxplots, mean (grey dotted line), and average TCP throughput (red dashed line) considering different run duration. Boxplots report 5^{th}, 25^{th}, 50^{th} (thick line), 75^{th}, and 95^{th} percentiles.

In the following section, I focus on the impact of the run duration on the difference between available bandwidth estimation and achieved TCP throughput. In the context of using available bandwidth as a proxy for TCP throughput, I refer to the difference between these two metrics as *accuracy*. It could be expected that an increased duration of measurement runs can provide higher accuracy. Therefore, I leverage the collected measurements, consisting in 20

second runs for both available bandwidth and achievable throughput, to validate this assumption and evaluate the accuracy on shorter time scales. The outcome of this analysis are reported in Figure 3.7, which includes the results of each run and for each country, considering a duration of 5, 10, 15, and 20 seconds. For Italy, I restrict the analysis on experiments leveraging 4G networks.

Based on the figure, different results can be highlighted according to the country under analysis. In detail, in Italy, the duration has a small impact on median value and variation of the available bandwidth estimates, while average throughput increases over time, as discussed earlier, and the average value lies below the estimated bandwidth during the first 5 seconds. In Spain, instead, as reported in Fig. 3.7b, variation of available bandwidth estimates decreases slightly after 5 seconds; at the same time, median estimation after 10 seconds are farther from the average achieved rate, showing opposing trends. Results for Norway (Fig. 3.7d) show a slight decrease in the variation for larger duration, while median value is stable and very close to the mean one, and about 2 Mbps above the average throughput. Instead, Sweden (Fig. 3.7c) is the only country not reporting significant changes in median of variation of available bandwidth according to the duration, while average throughput show slight increases.

Considering these results, one not intuitive finding is that the duration of a run does not necessarily improve accuracy in terms of relationship between available bandwidth and achieved throughput. This is more evident in the cases where throughput shows transient behavior, for example increasing for a non negligible amount of time before stabilizing. In these cases, the accuracy is higher when considering a 10-second interval. For this reason, estimates of

available bandwidth can be effectively used as proxy for TCP throughput when taking into account this transitory. When considering larger time intervals, one possible solution to address the discrepancies reported in some countries and operators would be to conduct a prior characterization of the time evolution of throughput, and correct the subsequent available bandwidth estimates according to these results.

3.4.5 Evaluating setup impact

Before delving into the details of the final experimental campaign conducted, here I refer to the experimental results from the long-term campaign to highlight some technical aspects regarding the tool used, Yaz, to further motivate (a posteriori) its use, showing the advantages compared to Pathload. Indeed, I report the stream compressions percentages, meaning, the number of streams (i.e. a sequence of packets, where multiple streams sent in sequence are also denoted as fleet) in which the mean spacing measured at the receiver side is lower than the sent one. Indeed, from a technical standpoint Yaz also considers spacing compressions, in addition to expansion, as evidence of congestion and adapts its behavior accordingly. Pathload, instead, does not employ such mechanisms, meaning that when received spacings of packets for a stream are lower than to transmitted ones, the tool considers the generated rate being below the available bandwidth.

Table 3.4 reports the average percentage of streams which encountered compression considering all the tests for each country. As can be seen, Spain and Italy are the country with the highest percentage of compressed streams. Recalling results of the preliminary experiments, I highlight that, not incidentally, Pathload performed worse in these specific countries, producing overes-

Table 3.4: Percentage of stream compressions during Yaz estimates in different countries.

Country	% of streams w. compressions
Norway	16.89
Spain	40.44
Italy-LTE	24.30
Italy-3G	29.09
Sweden	22.54

timates of the achievable throughput(in addition to high variability). I highlight that indeed similar percentages of stream compressions were recorded during the preliminary tests, with Spain reporting the highest percentage of compressions, and at the same time obtaining the highest error (overestimating throughput) for Pathload.

In addition to evaluating Yaz compression mechanisms on Mobile Networks, I also took into account the virtualized setup in which the experiments are executed on the MONROE platform, where each experiment must run within a Docker container. Although containers are a lightweight form of virtualization compared to full virtual machines, the limited hardware characteristics of the node required a preliminary investigation of the traffic capability in virtualized environments, as discussed in previous sections. Here, I quantify the impact of this virtualized setup referring to the results of the long-term campaign.

In detail, I focus on the capability of properly generating traffic according to inter-packet spacings requested by the application, since this parame-

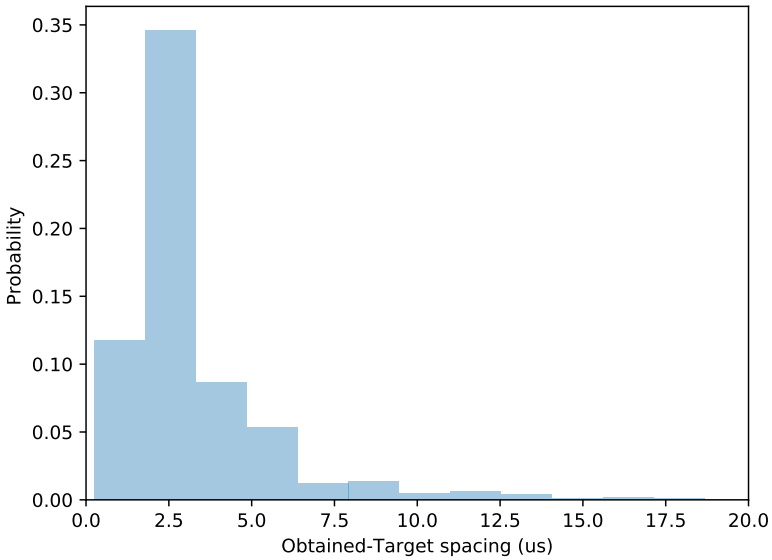


Figure 3.8: Distribution of differences between local and target spacings.

ter is crucial for the execution flow of available bandwidth estimation tools, therefore impacting on inter-arrival times and eventually on the bandwidth estimates. In Figure 3.8 I report the histogram of the differences between requested packet spacings and the generated ones. These differences do not take into account the additional overhead necessary on the host to transmit packets from the container to the physical network. From Figure 3.8 it can be seen that the difference is restricted to a very narrow range, with most of the differences being under 10 microseconds. This means that the use of virtualization is not expected to have a significant impact on the estimates, and that the variation of the inter-arrival times observed on the receiver can be ascribed to the network conditions, with the radio access network link representing the bottleneck in the considered mobile scenario.

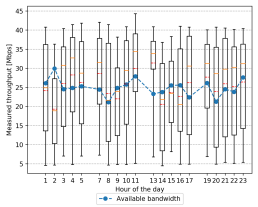
3.4.6 Whole-day experiments

The results of the longer campaign, previously discussed, have highlighted the need to observe TCP throughput in different conditions, in contrast with the setup discussed before aiming at minimizing cross-traffic and variability. Therefore, an additional campaign was designed in order to answer the following questions:

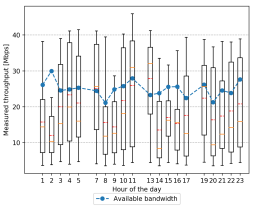
- Does throughput behavior depend on specific countries or providers?
- Does this behavior depend on the hour of the day?
- Does this behavior depend on the day of the week?

In detail, in this additional campaign (also reported in previous Table 3.1) I have performed one hourly test on two different nodes, running in sequence achievable throughput and available bandwidth tests as in previous campaigns, but reducing to duration of such tests to 10 seconds, according to results discussed in Sec. 3.4.4, primarily to reduce the volume of traffic generated without impacting the results. These hourly experiments were executed for 9 days (non-consecutive, but covering days for a whole week), over two nodes located in Sweden, served by two different providers.

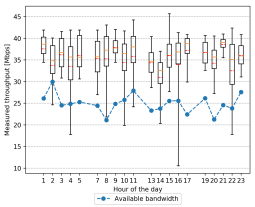
In Figure 3.9 I report the time-series of throughput and estimated available bandwidth focusing on two out of the three days of the whole-day campaign considering the two tested nodes. Results considering the first day are omitted since they exhibit similar behaviors. Missing values in the figure appear due to platform unavailability or deployment errors. The boxplots in the figures report the distribution of throughput sampled every 100 ms on intervals of different duration, to highlight the possible impact of traffic policies enforced by



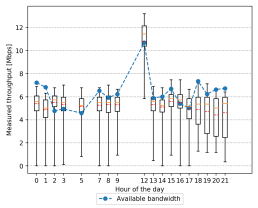
(a) Day 2. Node 1, 0–10 seconds interval.



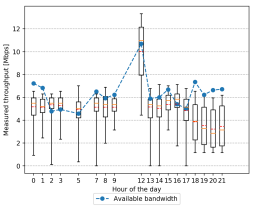
(b) Day 2. Node 1, 0–5 seconds interval.



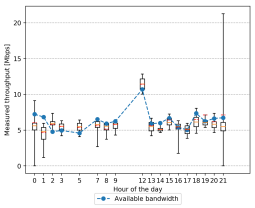
(c) Day 2. Node 1, 9–10 seconds interval.



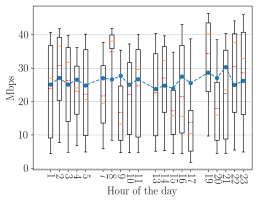
(d) Day 2. Node 2, 0–10 seconds interval.



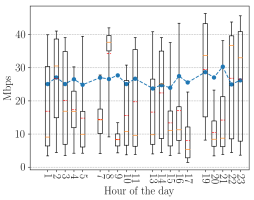
(e) Day 2. Node 2, 0–5 seconds interval.



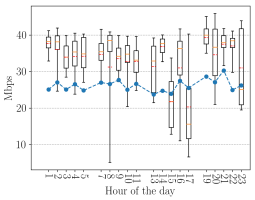
(f) Day 2. Node 2, 9–10 seconds interval.



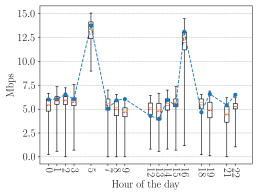
(g) Day 3. Node 1, 0–10 seconds interval.



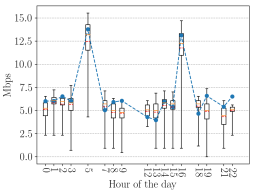
(h) Day 3. Node 1, 0–5 seconds interval.



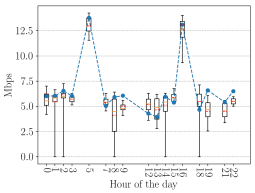
(i) Day 3. Node 1, 9–10 seconds interval.



(j) Day 3. Node 2, 0–10 seconds interval.



(k) Day 3. Node 2, 0–5 seconds interval.



(l) Day 3. Node 2, 9–10 seconds interval.

Figure 3.9: Available bandwidth (dashed line) and TCP achievable throughput (boxplots reporting 5^{th}, 25^{th}, 50^{th}, 75^{th}, and 95^{th} percentiles), considering different duration for measurement intervals during days 2 and 3 of the additional, whole-day experiments. Missing boxplots indicate node unavailability.

mobile operators during the first ten seconds of each throughput measurement, as discussed earlier. Therefore, the smallest interval, between 9–10 seconds, is related to the stable value reached after a transient phase.

Considering these results, higher variability appears for Node 1 (Fig 3.9g), as measured by the inter-quartile range. Available bandwidth and achievable throughput are close to 25 Mbps on average, considering the whole duration. Instead, Node 2 (Fig 3.9j) experiences limited variability, but also lower throughput (5 Mbps on average).

Considering throughput variation over different time intervals, possibly related to traffic policies, evaluation on shorter intervals for Node 1 exhibits higher difference compared to 10-second measurements; instead, results for Node 2 do not highlight major differences between intervals. This suggest that, in presence of a behavior similar to Node 2 (i.e. enforced policies do not limit throughput in the initial phase), available bandwidth tests on short scales can provide accurate results in representing stable value of achievable throughput. Instead, in the case of behaviors similar to that of Node 1, there is a constant bias which has to be taken into account given the difference between available bandwidth and throughput after the initial transitory period.

Comparing the two days of the campaign reported, it can be seen how behavior for the two nodes is consistent, with higher stability for Node 2 and higher, but also more variable, bandwidth reported for Node 1. Moreover, specific patterns cannot be devised, looking for example at the different times where spikes for Node 2 appear during the two days.

Therefore, considering the initial questions leading to the additional experimental campaign, results have highlighted that the throughput behavior depends on the provider, but does not vary over time, meaning it does not ex-

hibit behavior related to either the hour of the day or the day of the week. This finding has a direct impact on the applications. For example, this means that when characterizing throughput to correct available bandwidth estimates, it can be reasonably expected that the specific time interval considered for the experiments does not alter the accuracy of these measurements on different ranges, having a general validity.

3.5 Remarks and discussion

In this chapter, I have considered active bandwidth estimation in mobile networks, which are typically more constrained compared to wired setups. Leveraging an European research testbed, I have conducted extensive experimentation, testing different operators in different countries over Europe. In detail, I have considered two bandwidth related metrics, namely available bandwidth and achievable throughput, and assessed the use of the first as a proxy for the second, achieving comparable accuracy at a reduced cost in terms of generated traffic volume.

The ability of accurately monitoring bandwidth is crucial considering the QoS requirements of the applications considered in this thesis and detailed in Chapter 1. For example, multimedia applications such as video streaming often adapt their behavior to match the network conditions, employing adaptive bitrate strategies. This applies to several telemedicine applications as well, for example in the case of remote imaging applications, where the bandwidth requirement is essential to ensure high-quality images for diagnosis, or for the remote computation use case to reduce the transfer time of images to the remote servers. In non-real time use cases, bandwidth monitoring still helps ensuring

that the minimum bandwidth requirements are satisfied, allowing healthcare services to operate reliably.

While the approach analyzed in this chapter aims at minimizing intrusiveness, it still works by injecting probe traffic into the monitored network (i.e. traffic crafted with the only purpose of monitoring), and therefore classifies as an *active* bandwidth estimation approach. In cases when fine-grained real-time monitoring is required, *passive* approaches may be preferred as they do not pose additional traffic into the network, which may possibly impact application performance, especially in high load conditions. For this reason, the next chapter is focused on a passive, SDN-based approach for bandwidth monitoring.

Chapter 4

Passive bandwidth estimation in mobile networks leveraging SDN

After evaluating available bandwidth as a proxy for achievable throughput using active methods, in this chapter I introduce and evaluate the accuracy of a passive (i.e. not intrusive) bandwidth estimation method developed previously [72], which leverages the possibilities offered by the rising Software-Defined Network architectures. To this goal, I have conducted an experimental campaign on commercial mobile networks, leveraging the MONROE platform, as in the case of active bandwidth estimation.

In the previous chapter, I have already highlighted the importance of mobile networks and its performance for novel applications. The transition towards the Fifth Generation of mobile communications (5G), which has already begun, enables a fully mobile and connected society and brings several infras-

tructural innovations [9]. Among the most important, a massive deployment of Mobile Broad Band networks, far increasing the coverage already established with 4G networks, which has less strict requirements on bandwidth and latency. The benefits of this scenario also lead to an increased complexity of the network architectures, in contrast with the need of quickly adapting them to new emerging scenarios. For this reason increased importance was given to the recent network implementation and management paradigm known as Software Defined Networking (SDN), which has seen increasing adoption, especially in the data center context for their fast-paced evolution [27].

However, the transition to 5G has given more importance to mobile terminals that can become another application scenario benefiting from SDN. MBB access (4G) networks already demonstrate cases of access sharing between multiple devices, for example mobile hotspots or mobile wireless router (so-called *Mi-Fi*); at the same time, wireless networks can be used as a backhaul for smart cities [59, 60], and even vehicles nowadays are equipped with network applications for different goals, such as entertainment, traveling assistance, comfort, or maintenance. All these cases scenarios can be modeled as mobile nodes performing as gateways for different applications and devices (each with different requirements), all sharing a common Radio Access Network (RAN) when accessing the Internet.

This scenario can particularly benefit from the SDN paradigm, for its main feature of flexibility and standardization, fostering new monitoring approaches. Still, its novelty and the application scenarios characterized by resource-limited devices pose challenges and issues to the adoption of SDN in MBB, which are an active research topic requiring experimental evaluation and analysis leveraging realistic testbeds (or in-the-wild deployments).

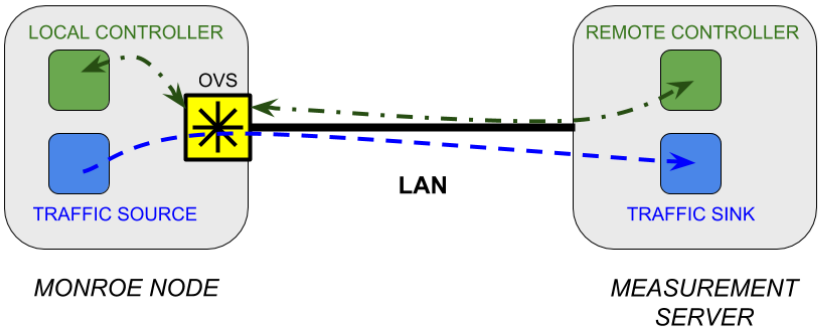
The possibilities for experimenters to access such testbeds was the goal of the MONROE project, which, as discussed in the previous chapter, is an European Union's Horizon 2020 funded research project, in which the SOMETIME project was funded in our research group. The SOMETIME project (SOftware defined network-based available Bandwidth MEasuremenT In MONROE) was part of the 1^{st} MONROE Open Call for Experiments. Leveraging the MBB access networks offered by the platform, the SOMETIME experiment focused on Available Bandwidth estimation in an SDN environment, testing commercial 4G connections.

In the work presented hereafter, I implement and evaluate a state-of-art SDN-based approach proposed in [72] for monitoring available bandwidth and throughput in the real-world MBB scenarios provided by the MONROE platform. In detail, the investigated approach leverages the network abstractions provided by the SDN southbound interface and in particular is designed to run on the SDN controller, which queries all the switches deployed in the mobile network periodically to collect counters and estimate bandwidth according to a passive approach. I remark that this passive approach contrasts with the active, and therefore intrusive and expensive one, evaluated in the previous section, which however does not require infrastructural changes to deploy an SDN network, and thus can be implemented quicker and with less expenses. Therefore, the two approaches can be seen as complementary, as detailed in the concluding remarks of this chapter.

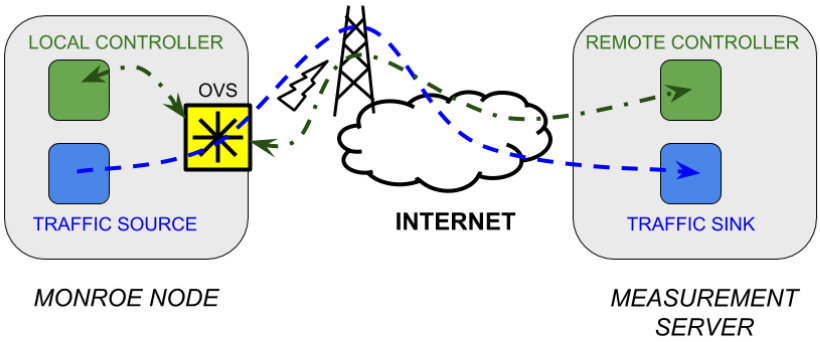
Summarizing, in the work described in this chapter, I conducted what was, at the time of the experiments and to the best of my knowledge, the first experimental evaluation of an SDN-based bandwidth estimation method in a commercial MBB scenario, evaluating different setups, including a local and a

remote SDN controller. These are compared with an analogous fully wired setup, as shown in Figure 4.1 and detailed later. Considering the novelty of the approach illustrated in this chapter, first, compared to [72], I target in-the-wild mobile networks, while the cited work has only considered an emulated SDN network for the evaluation of the same bandwidth estimation method here considered. Instead, Van Adrichem et al. [119] have considered a physical setup leveraging a controlled testbed, thus ignoring the sources of interference of real networks. Moreover, they have also neglected the impact of sampling frequency and controller position with respect to switch on the results, while I have instead included these as experimental parameters. Indeed, I assess the accuracy of such monitoring approach when the polling period of the queries made by the controller changes, in addition to the deployment aspect represented by the SDN controller position, and the amount of traffic to be monitored in terms of average throughput. From the results it appears that high accuracy can be reached on average, with mean error close to zero in all the investigated scenarios. While this is true considering the average results, the polling period in particular influences the variability of punctual measurements.

In the rest of the chapter, I first provide background for SDN in Section 4.1, as it was not considered in the previous, active bandwidth estimation approach. The working principle of this method is summarized in Section 4.2; within it I also discuss the details of the experiment design and the measurement scenario. Then, I report the experimental analysis in Section 4.3, discussing the main outcomes. Finally, in Section 4.4 I provide concluding remarks to the passive method and, more generally, to the bandwidth estimation approaches presented in this thesis, considering advantages and disadvantages of both of them.



(a) Wired LAN deployment.



(b) RAN/Internet-crossing deployment.

Figure 4.1: Details about the measurement setups considered for SDN-based passive bandwidth estimation. The dashed blue lines report the application traffic, while dotted-dashed green lines represent the OpenFlow messages exchanged between the controller and the switch, for both the *local* and *remote* controller deployments.

4.1 Background

Here I provide background to the working principle of the investigated approach, detailing the SDN paradigm. I instead refer to the previous Section 3.1

to recall the available bandwidth and throughput definitions and for details about the MONROE project providing the mobile network testbed for the experiments here reported.

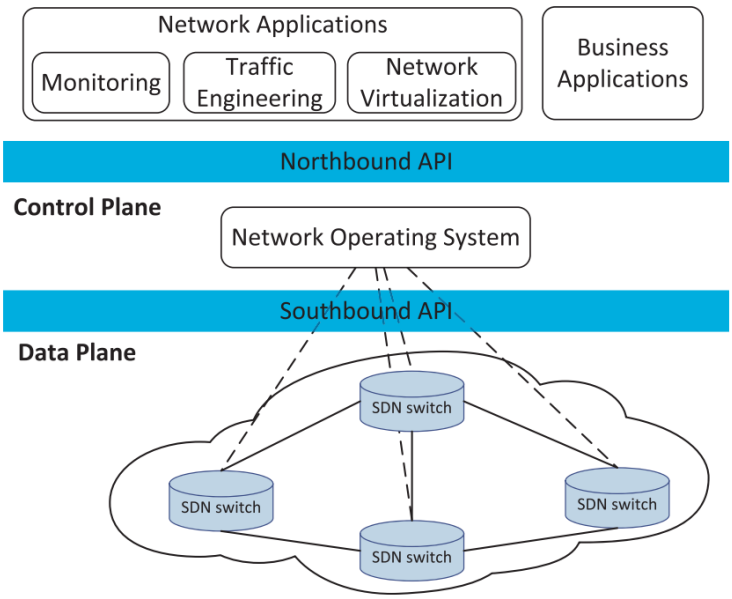


Figure 4.2: Architecture of Software Defined Networks (source [72]).

Software-Defined Networking Software-Defined Networking (SDN) is a recent paradigm for network devices management, following previous works dealing with active networks and network virtualization. The main concept of SDN relies on the functional separation between control and data planes, as shown in Fig. 4.2. In this view, the network devices implementing the data plane are simple forwarding elements, and are named *SDN switches* or simply *switches*, whose behavior is set according to forwarding rules installed in their memory by an *SDN controller*, based on a set of fields in the packet

headers. Thus, the control plane is moved onto the external, and logically centralized controller. In this way, a unified behavior for the networking elements is achieved, regardless the high-level function and the protocol layer they act on (switches, routers, firewalls, etc.). The current importance to SDN was reached after the introduction of OpenFlow [71] as a standard protocol for controller-switch communication, denoted as *southbound interface* according the OpenFlow terminology, as opposed to the *northbound interface* denoting the communication between SDN controller and the application running on top of it. More details about SDN can be found in [62]. Network monitoring can benefit from SDN, as witnessed by several works. The most similar to the scenario I detail here is proposed by Van Adrichem et al. [119], who consider OpenFlow monitoring messages to calculate the flow throughput averaged on the whole flow duration. Other works indirectly exploit the control communications to (passively) infer network metrics [131], or adopt sampling [100]. While I refer to [72] for more details on related works, which often consider different metrics and adopt different approaches, up to my knowledge none of these works has considered bandwidth measurement with OpenFlow on MBB networks.

4.2 Methodology and setup

The measurement methodology, details about the experimental parameters taken into account and the experimental setup considered are discussed in the next Sections.

Table 4.1: Mean and standard deviation (%) for ΔT *relative error* in Wired-LAN deployment using *local* controller.

		Requested bitrate (Mbps)					
		0.1	**0.5**	**1**	**5**	**10**	**50**
Period (s)	**0.5**	0.0078 ± 0.4590	0.0076 ± 0.3586	0.0064 ± 0.5999	0.0035 ± 0.6755	0.0056 ± 1.6695	0.0039 ± 1.9629
	1	0.0079 ± 0.2505	0.0070 ± 0.2055	0.0067 ± 0.3315	0.0050 ± 0.4108	0.0070 ± 1.2465	0.0045 ± 1.3440
	2	0.0072 ± 0.1156	0.0067 ± 0.1121	0.0068 ± 0.1814	0.0054 ± 0.2313	0.0078 ± 0.7372	0.0070 ± 0.5825
	5	0.0060 ± 0.0500	0.0065 ± 0.0535	0.0064 ± 0.0755	0.0064 ± 0.1021	0.0076 ± 0.3062	0.0111 ± 0.3243
	10	0.0062 ± 0.0272	0.0062 ± 0.0303	0.0062 ± 0.0386	0.0066 ± 0.0522	0.0068 ± 0.1557	0.0116 ± 0.1712

Table 4.2: Mean and standard deviation (%) for ΔT *relative error* in Wired-LAN deployment using *remote* controller.

		Requested bitrate (Mbps)					
		0.1	**0.5**	**1**	**5**	**10**	**50**
Period (s)	**0.5**	0.0059 ± 0.1462	0.0106 ± 0.0992	0.0100 ± 0.1279	0.0066 ± 0.1749	0.0110 ± 0.2206	0.0071 ± 0.2157
	1	0.0069 ± 0.0841	0.0086 ± 0.0676	0.0091 ± 0.0824	0.0080 ± 0.0995	0.0104 ± 0.1196	0.0073 ± 0.1187
	2	0.0080 ± 0.0530	0.0077 ± 0.0379	0.0081 ± 0.0567	0.0081 ± 0.0540	0.0101 ± 0.0678	0.0093 ± 0.0669
	5	0.0066 ± 0.0251	0.0071 ± 0.0184	0.0073 ± 0.0248	0.0066 ± 0.0267	0.0086 ± 0.0281	0.0061 ± 0.0361
	10	0.0052 ± 0.0114	0.0049 ± 0.0106	0.0053 ± 0.0123	0.0043 ± 0.0110	0.0067 ± 0.0158	0.0035 ± 0.0170

4.2.1 Measurement methodology

The measurement method under investigation is based on the API exposed by SDN switches to collect information about the traffic flowing into the network. To this goal, the measurement application running on the top of the SDN controller queries the switches at different time instants for their volume counters, stored for each flow. Leveraging this counter, the controller obtains a global view of the traffic volume crossing different links. Therefore, by evaluating the

value of volume counters at two different points in time, the network throughput B_i on the link can be estimated as

$$B_i = \frac{V_i - V_{i-1}}{T_i - T_{i-1}} = \frac{\Delta V_i}{\Delta T_i} \tag{4.1}$$

where V_i and T_i are the data volume counter and the timestamp of the i-th query respectively. If the capacity of each link of a path is known, the available bandwidth on the link can be derived subtracting the estimated throughput from the capacity, and on the path as the minimum value among the links.

Specifically, since the considered method employs OpenFlow as the most widespread protocol for SDN, queries consist of `FlowStats Request` messages, while `FlowStats Reply` messages report the volume counters to the controller. For the detailed message exchange during the measurement process I refer to [72]. According to the most recent standard version at the time of experiments, (OF 1.5), it should be noted that OpenFlow does not store the *actual* timestamp associated to volume counters in the messages. However, reliable timestamps are crucial for the controller in order to reduce the error: in fact, errors in evaluating ΔT_i may significantly impact the accuracy of the throughput estimate. I also refer to [72] for a more comprehensive analysis and also possible extensions to the OpenFlow protocol to solve this problem. In the experimental work performed, I evaluate the impact of two important parameters, related to the measurement process as well as to the deployment of the network components: (i) the polling period and (ii) the position of the controller with respect to the switch. The polling period represents the time distance between two consecutive `FlowStats Request` messages: lower periods allow to obtain finer granularity, but at the same time it amplifies the impact of errors on measurement accuracy and uncertainties of the timestamp-

ing process. Concerning the controller placement in the network, this can be necessary for architectural and deployment constraints, and therefore I remark that the impact of this parameter should be investigated in detail.

4.2.2 Experiment design

In the experimental campaign conducted I have considered the following values for the experimental variables introduced earlier: (i) local or remote placement of the controller; (ii) 6 different rates for the constant bitrate traffic flows generated by D-ITG [17, 31] (0.1 Mbit/s, 0.5 Mbit/s, 1 Mbit/s, 5 Mbit/s, 10 Mbit/s and 50 Mbit/s); (iii) and 5 different polling periods at which the controller queries the information from the switch (0.5 s, 1 s, 2 s, 5 s, 10 s). I underline that this extended set of parameters was not considered in the experimental evaluation of any of the previously cited works, which adds to the fact that I target in-the-wild mobile networks. Results shown in Section 4.3 are all obtained starting from a polling period of 0.5 s. This choice is motivated by a two-hour preliminary campaign where I evaluated different periodicity for the controller queries, and chose to reconstruct results related to larger polling periods by sampling the results obtained with the 0.5 s polling period with different rates (i.e. $1/2$, $1/4$, $1/10$, $1/20$). I remark that this design choice allows to obtain a more fair comparison, without introducing additional bias by performing additional experiments, due to for example network variability, since the experiments are conducted on a commercial 4G network. For each scenario presented the the obtained statistics refer to 100 samples. In these experiments, the application on top of the controller monitored the throughput (using OpenFlow `FlowStats Request`/`FlowStats Reply` message exchange), while D-ITG was used to generate background traffic. Moreover,

the traffic exchanged between the switch and the controller is captured on the switch, in order to perform a comparison between the inter-departure times of the replies sent by the switch (which constitute the ground truth when assessing the accuracy), with the actual ΔT seen on the controller side, considering the timestamps associated to the `FlowStats Reply` messages. The accuracy is then evaluated by considering the ΔT relative error as the difference between the interval estimated by the controller minus the interval seen on the switch, normalized to the ground truth (i.e. the interval on switch side). In the next sections, I report this value in percentage. This error has a direct relationship with the error in estimating network throughput. Indeed, throughput is calculated considering difference in the counter values returned by the switch divided by the time interval between the two replies. As there is no additional error or degradation associated to the traffic counters, the only factor affecting the accuracy is the timestamp error, and throughput and available bandwidth are linearly (inversely) dependent on ΔT estimation error only, according to equation 4.1.

4.2.3 Measurement scenario

I consider two different scenarios, the first one employing a wired connection to a local server, and the other crossing the Radio Access Network to a remote server, as shown in Figure 4.1. In both cases the end hosts are configured at the same way, but are connected differently. In the first case, as seen in Figure 4.1a, I leverage the the mobile node deployed in our laboratory at the University of Napoli, which is directly connected to the measurement server through a 100 Mbps Ethernet LAN. In the second case, reported in Figure 4.1b, the measurement server is the same as before, while I leverage to MONROE platform to

Table 4.3: Hardware and software characteristics of the mobile node and the measurement server.

HW/SW spec.	Measurement server	Mobile node
CPU	i7-4710MQ @ 2.50GHz x8	AMD G-T40E @ 1 GHz x2
RAM	12 GB DDR3 @ 1600 MHz	4 GB DDR3 @ 1066 MHz
NIC	Gigabit Ethernet	3xGigabit Ethernet ports
4G	—	3xZTE MF910 MiFi
OS	Ubuntu 16.04 64 bit	Debian-Jessie
Kernel	4.4.0-119-generic	4.9.0-6-amd64

gain access to a mobile node deployed in Karlstad (Sweden), connected to the Internet through a 4G access network, served by Telia operator. In both cases: (i) the software OpenFlow switch (namely Open vSwitch, or OVS for short) is deployed onto the mobile node, and is configured to serve as gateway for all the inbound and outbound traffic for the node; (ii) DITG [17, 32] is used as synthetic traffic generator on the mobile node in order to generate constant-bitrate traffic at the different rates reported earlier towards the measurement server; (iii) Moreover, I have considered experiments with the SDN controller (Ryu, an open-source controller written in Python) placed either on the mobile node (I refer to this scenario as *local controller*) and on the measurement server (*remote controller* scenario). As in all experiments running onto the MONROE platform, the software on the mobile node runs inside a lightweight virtualization environment in the form of a Docker container [30]. I also detail the hardware and software characteristics of mobile node and measurement server in Table 4.3.

4.3 Experimental results

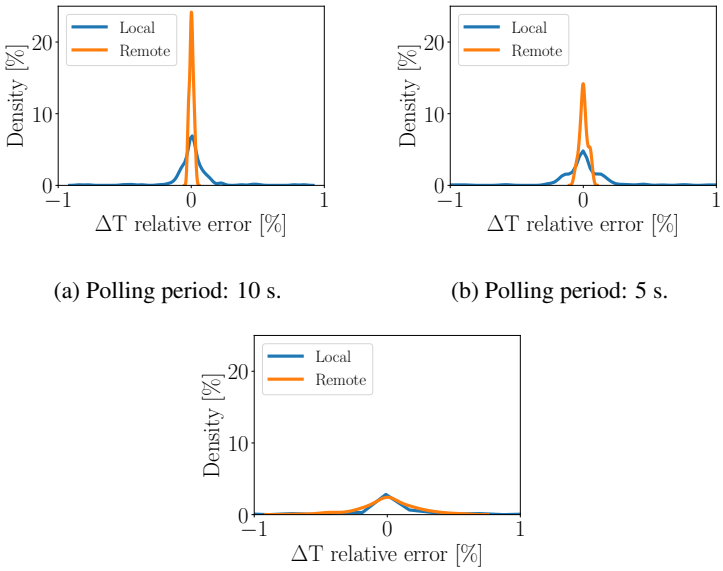
Considering the results for all the previously reported scenarios, it was assessed that the relative error on ΔT is most of the time concentrated around zero, and also reports a small variability, with a standard deviation is smaller than 1%. I discuss these results more in detail, considering the wired and the RAN scenarios separately. In the following I detail the results of the experimental campaigns, first for the wired-LAN deployment and then for the RAN/Internet-crossing one, discussing edge cases.

Wired LAN deployment

Results for the wired LAN deployment are detailed in Tables 4.1 and 4.2 reporting mean and standard deviation of the relative error for local and remote controller, respectively, and considering the different combinations of polling period and traffic rate, as discussed before. For both the local and remote controller the average relative error is always below 0.015%. Considering the distribution, instead, as reported in Figure 4.3, a symmetric behavior can be seen, with lower periods leading to higher variability, and reflected by an increased standard deviation. A similar results is obtained when increasing the traffic bitrate, particularly for the local controller. This likely happens because of the resource sharing between the synthetic traffic generator and the local controller that also runs on the switch, inside the Docker container.

RAN/Internet-crossing deployment

I now focus on the results considering the RAN deployment. Figure 4.4 reports the relative error of ΔT, for different polling periods (on different rows)



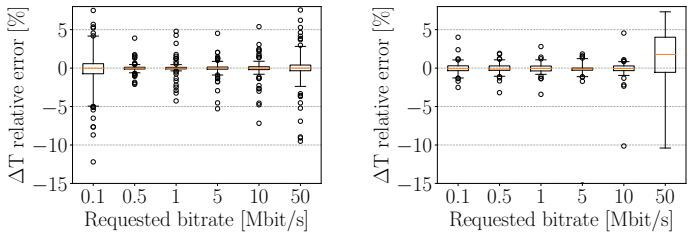
(a) Polling period: 10 s.

(b) Polling period: 5 s.

(c) Polling period: 0.5 s.

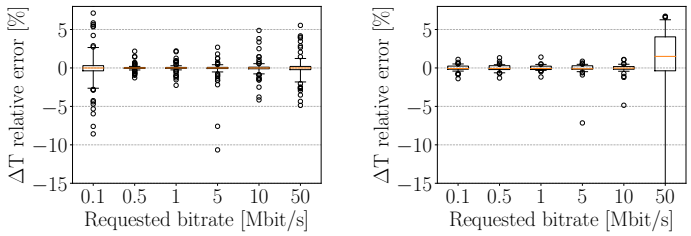
Figure 4.3: Probability density function (PDF) for the relative error in the wired-LAN deployment, considering local and remote controller deployment and different polling periods. Background D-ITG traffic is generated at 50 Mbit/s. Results show that when using longer polling periods the error distributions are more concentrated around the mean.

and considering again local and remote controller (left and right column, respectively). The boxplots report 5^{th} percentile, 25^{th} percentile, median, 75^{th} percentile, and 95^{th} percentile, while values below and above the 5^{th} and 95^{th} percentiles are represented as outliers (black circles). In the case of the local controller (Figs. 4.4a, 4.4c, 4.4e, 4.4g, 4.4i) there is a higher variability when the rate of the requested traffic is higher; the only exception to this behavior is represented by the lowest bitrate (0.1Mbps), which instead shows higher vari-



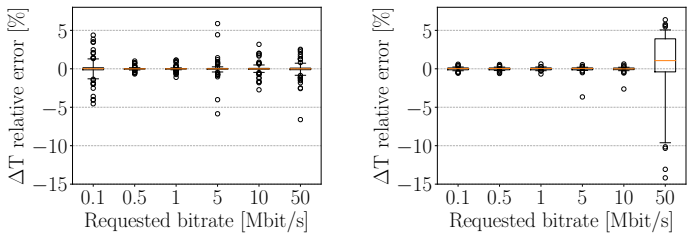
(a) Polling period: 0.5 s.

(b) Polling period: 0.5 s.



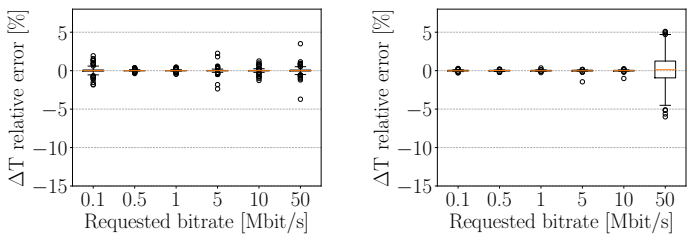
(c) Polling period: 1 s.

(d) Polling period: 1 s.



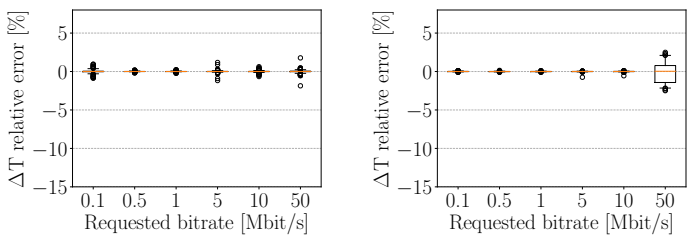
(e) Polling period: 2 s.

(f) Polling period: 2 s.



(g) Polling period: 5 s.

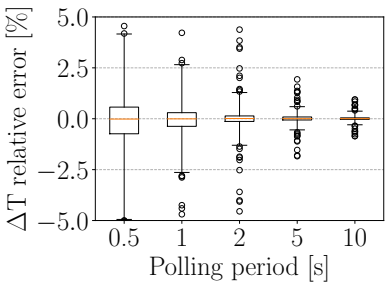
(h) Polling period: 5 s.



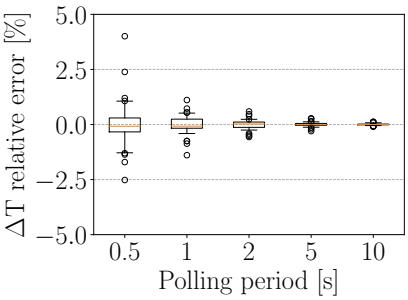
(i) Polling period: 10 s.

(j) Polling period: 10 s.

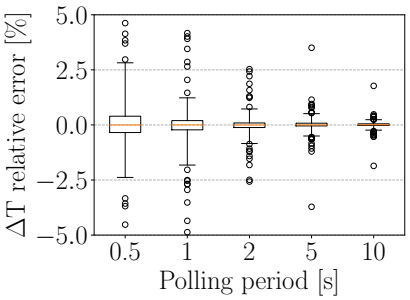
Figure 4.4: Relative error with different requested bitrates, in the case of a local (left) and remote (right) controller in the RAN deployment.



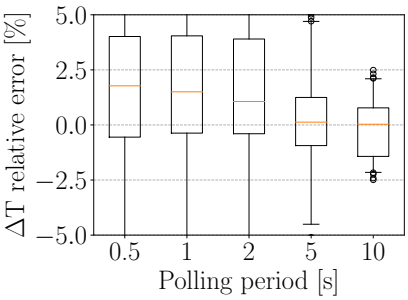
(a) Requested rate: 0.1 Mbps.



(b) Requested rate: 0.1 Mbps.



(c) Requested rate: 50 Mbps.



(d) Requested rate: 50 Mbps.

Figure 4.5: Relative error with variable polling periods in the case of a local (left) and remote (right) controller in the RAN deployment. The range of the Y-axes is restricted so as to magnify the differences.

ability. For possible explanations for this behavior, I once again remark that the experimental scenario includes a virtualized environment, that at the same time runs the SDN controller, the switch, and the traffic generator, and each of these can be therefore affected by scheduling policy, queuing of node-local communications and issues related to the hypervisor and the physical hosts.

The results for the remote controller are instead shown in Figs. 4.4b, 4.4d, 4.4f, 4.4h, 4.4j. In this case, variability in terms of inter-quartile range ex-

hibited by the boxplots is confined and does not vary significantly with the increasing requested traffic rate. Again, one exception appears, but this time related to the highest requested rate (50Mbps): in this case, it can be seen that variability increases more than linearly with respect to the case of 10Mbps, and also 5^{th} and 95^{th} percentiles are farther apart, crossing $+5\%$ and -10% relative error levels when considering a 0.5 s period. The reason of this phenomenon, which can be seen constantly across the different polling periods, is that replies from switch to controller share the same link used by the background traffic to be monitored. Therefore, when a higher rate is requested, the bandwidth of the radio link is saturated, and so the delays grow significantly, directly impacting the received timestamps which, as discussed, are the main cause of error in estimating bandwidth with this approach. The saturated link was also confirmed using the logs produced by D-ITG, which report that, although the requested rate was indeed of 50 Mbps, the actual achieved throughput was less than a half.

These results highlight that the deployment aspect of the controller indeed impacts significantly on the accuracy when estimating bandwidth, especially if switch and controller communication happens in-band (i.e. using the same network to be monitored) and the load is higher.

A complementary view made to emphasize the effect of polling period given a requested background traffic rate is given in Figs. 4.5a–4.5d. This Figure only reports a representative subset of all the periods, since the observed behavior was consistent in the remaining cases. It can be seen that the error variability of the relative error decreases with the increasing polling period, as expected and discussed earlier, for both local and remote controller. Indeed, when using a higher polling period the uncertainty deriving from the

timestamping process is averaged over a wider interval, and therefore becomes negligible. The higher accuracy however is achieved by reducing the monitoring frequency, and therefore impacts on the ability to quickly detect and respond to sudden changes in the traffic volume, for example considering traffic spikes which can appear especially in specific hours of the day. Moreover, a higher polling period also requires less computational resources, and this aspect should be taken into account, especially in the case of the local controller, as previous results have highlighted.

4.3.1 Results discussion

From the results shown in the previous sections, it appears that the passive approach to measure bandwidth on an SDN switch co-located with a mobile terminal is feasible, since it reports a mean relative error close to zero, and a bounded standard deviation. Extending this analysis to the error distribution, I remark that each punctual estimate suffers from an error, which can be more significant in the case of high traffic rates and low polling period; this error can be reduced leveraging multiple subsequent measurements, but losing the ability to quickly respond to anomalous events.

I recall that in order to derive the available bandwidth, the link capacity of the radio access link is needed. This information can be acquired through, for example, device-provided link status data [128], or by active measurements targeted at capacity estimation [3]. As seen when discussing active available bandwidth estimation methods, in both cases there are sources of error which have to be taken into account. From throughput measurements, available bandwidth can be derived also by considering the nominal capacity of a given transmission technology, which can be also obtained leveraging mobile

device metadata [128], which were available on the MONROE platform. This metadata of course only constitute an upper bound, and can be for example used to enforce an optimistic admission control.

The experimental scenario here presented represents a more complex one, with the OpenFlow switch located on the mobile node and acting as a gateway for the radio access link, therefore representing the monitoring and management point of the local network. The inbound and outbound traffic is emulated using a synthetic network traffic generator, not lacking of generality for the SDN-based measurement considered in this work. Indeed, the aggregate volume of diverse traffic is accounted for at the same way, and without any measurement error, by the SDN switch.

Instead, having the possibility to program the local SDN network behavior in response to changes in the RAN link bandwidth allows for novel applications, including admission control and traffic engineering, taking into account the bandwidth requirements of each application and programming the network to satisfy them.

4.4 Remarks and discussion

In this chapter, I have detailed and shown the experimental results related to a passive SDN-based approach for available bandwidth and throughput estimation on 4G Mobile Broadband Networks. In detail, I have shown the sources of inaccuracy and the main parameters affecting it, considering the of the controller deployment (local to the mobile node, or remote deployed on the other end of a path comprising a Radio Access link), and also the impact of variable polling period and traffic conditions. These results were compared with a fully

controlled setup where source and destination node are linked by a wired LAN connection, to emphasize the effect of network conditions variability on the bandwidth estimation approach.

The results have confirmed that indeed the approach is viable also in mobile nodes accessing the network through a radio link, without requiring significant resource to provide accurate results, and impacting the results only in the specific case of the SDN controller deployed on the mobile node together with the SDN switch and the traffic generator, all inside a Docker container. In all the cases the relative error on the time interval estimation is close to zero on average, and its variability in terms of standard deviation ranges between 1.31 and 8.65%, with highest values, as reported earlier, obtained when the requested bitrate saturates the mobile link bandwidth and the controller is remote. These results motivate possible further research, investigating for example the impact of the path connecting switch and controller not only in terms of bandwidth but also considering other QoS parameters as latency, jitter, or packet loss, with the ultimate goal of developing a model that takes all of these factors into account to improve the accuracy. Finally, considering both the passive SDN based approach presented here and the active available bandwidth estimation presented in Chapter 3, clearly both approaches have their advantages and disadvantages. Indeed, the main disadvantage of the active approaches lies in in their intrusiveness, as traffic injected into the monitored network can possibly interfere with other applications; this is especially true when the considered network is more constrained in terms of bandwidth, or when the load is high. At the same time, the active approach described in Chapter 3 does not require the deployment of an SDN network, being easier to implement and evaluate on a wider range of scenarios. While passive ap-

proaches can be applied also considering non-SDN setups, in these cases their results depend on the traffic flowing into the network at a given time, as typically these approaches leverage existing traffic to infer the metrics of interest.

Similarly, the presented passive SDN-based approach is not intrusive, but as shown it can suffer from accuracy errors that also depend on the traffic load in the network. This aspect can be characterized before starting the measurement application, but, especially in mobile networks, characterizing network load can be harder. Therefore, a natural conclusion is that the two approaches are not mutually exclusive, preferring one over another according to the specific scenario, but can also be used in conjunction. For example, depending on the degree of accuracy required and from the load of the network, one can choose to adopt an active probing approach to complement or replace passive measurements.

Conclusions

In this thesis, I considered the demanding requirements of latency and bandwidth sensitive applications, such as telemedicine ones, and the possibilities offered by cloud computing infrastructures and mobile scenarios, that however have also different drawbacks that need to be tackled, mainly concerning monitoring network performance, a key element to ensure a widespread adoption. These aspects were detailed in Chapter 1. Therefore, in the work presented I focus on latency and bandwidth, two metrics whose predictable performance is of paramount importance in this context, and which require the design and the evaluation of proper monitoring methodologies.

To this goal, first, in Chapter 2, I conducted an in-the-wild assessment of cloud-to-user network performance considering the two main public cloud providers, AWS and Azure. Through a 14-days campaign leveraging 25 Vantage Points scattered all over the globe, and targeting 4 different Cloud Regions for each provider, I provided a detailed characterization of latency in this context. I also demonstrated how the results can support cloud providers and customers in deploying their infrastructure and their applications, and help them troubleshooting in case of anomalous events.

Then, in Chapter 3 I focused on active bandwidth estimation approaches in

Mobile Broadband networks, leveraging a real testbed provided in the scope of the MONROE project, and assessing in which conditions the available bandwidth can be used effectively as a proxy for TCP throughput. Testing several mobile network operators in 4 different countries, I found that available bandwidth can provide good accuracy in estimating TCP throughput while requiring a significant lower traffic volume. I also found that the accuracy is impacted by traffic policies implemented by operators, especially when throughput exhibits transient behavior. In this case, the available bandwidth is closer to throughput during this transient phase, and therefore proper offset is needed to estimate stable throughput.

Finally, in Chapter 4 I focused on a passive bandwidth estimation approach, leveraging the possibilities offered by the Software Defined Networks paradigm. Again testing commercial Mobile Broadband networks, I assessed the accuracy of such techniques when the traffic load on the network and the polling period at which bandwidth is monitored vary, also the evaluating the impact of the controller position on the results. As main findings, I found that traffic load can visibly impact the estimates when the load is higher, polling period is short and the controller communicates with the switch remotely through an Internet connection including a mobile hop. The reason for this behavior is the in-band communication between switch and controller suffering from the high load. In all the other tested conditions the approach is feasible, with mean error close to zero.

In addition to the results discussed, the work conducted has also traced the way for future work, as detailed in the following.

Extending the cloud latency dataset. While the collected dataset already consists of a considerable number of Vantage Points, higher than other state-of-the-art approaches having the same goals, additional data can be collected to expand the overall dataset, providing more up-to-date samples, leveraging additional Cloud Regions, and targeting the edge infrastructures being increasingly deployed by several providers (for example considering the CloudFront edge locations offered by AWS).

Prediction of cloud performance. With the collection of an extensive dataset consisting in C2U network latency samples, possibilities open up to leverage prediction techniques to forecast latency in the cloud scenario, allowing to perform proactive management of the infrastructure. For example, switching from one provider to another in multi-cloud deployments according to which one is expected to provide better performance, or developing a less-invasive probing process using adaptive probing techniques, thus saving bandwidth and computational resources. To this goal, state-of-the-art deep learning architectures can be leveraged, for example exploiting the capabilities of recurrent architectures commonly used for time-series prediction, or evaluating transfer learning methodologies to exploit previously collected data or also data concerning other Cloud Regions or different protocols, to help the prediction in different (but related) contexts.

Hybrid bandwidth estimation approaches. I have separately evaluated an active and a passive bandwidth estimation approach, and as they both have their advantages and drawbacks, the design of an hybrid methodology could exploit the benefits of both, for example leveraging passive approaches when

the traffic load on the monitored network and the requested probing period allow to achieve a good accuracy, using active approaches otherwise, or combining the two techniques to achieve a higher accuracy.

Bibliography

[1] A. Abou El Kalam, A. Ferreira, and F. Kratz. Bilateral teleoperation system using qos and secure communication networks for telemedicine applications. *IEEE Systems Journal*, 10(2):709–720, 2016.

[2] Giuseppe Aceto, Fabio Palumbo, Valerio Persico, and Antonio Pescapé. An experimental evaluation of the impact of heterogeneous scenarios and virtualization on the available bandwidth estimation tools. In *Measurement and Networking (M&N), 2017 IEEE International Workshop on*, pages 1–6. IEEE, 2017.

[3] Giuseppe Aceto, Valerio Persico, Antonio Pescapé, and Giorgio Ventre. Sometime: Software defined network-based available bandwidth measurement in monroe. In *Network Traffic Measurement and Analysis Conference (TMA), 2017*, pages 1–6. IEEE, 2017.

[4] Giuseppe Aceto, Fabio Palumbo, Valerio Persico, Haiming Chen, and Antonio Pescape. Evaluation of sdn-based bandwidth estimation in mobile broad band networks. In *2018 24th Asia-Pacific Conference on Communications (APCC)*, pages 263–268, 2018.

[5] Zahid Akhtar and Tiago H. Falk. Audio-visual multimedia quality as-

sessment: A comprehensive survey. *IEEE Access*, 5:21090–21117, 2017.

[6] Shaimaa Al-Janabi, André Huisman, and Paul J Van Diest. Digital pathology: current status and future perspectives. *Histopathology*, 61 (1):1–9, 2012.

[7] Özgü Alay, Andra Lutu, Rafael García, Miguel Peón-Quirós, Vincenzo Mancuso, and et al. Measuring and assessing mobile broadband networks with monroe. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–3, June 2016.

[8] Ahmed Ait Ali, Fabien Michaut, and Francis Lepage. End-to-end available bandwidth measurement tools : A comparative evaluation of performances. *CoRR*, abs/0706.4004, 2007.

[9] A. Annunziato. 5g vision: Ngmn - 5g initiative. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5, May 2015. doi: 10.1109/VTCSpring.2015.7145586.

[10] E. Ataie, R. Entezari-Maleki, L. Rashidi, K. S. Trivedi, D. Ardagna, and A. Movaghar. Hierarchical stochastic models for performance, availability, and power consumption analysis of iaas clouds. *IEEE Transactions on Cloud Computing*, 7(4):1039–1056, 2019.

[11] Vaibhav Bajpai, Anna Brunstrom, Anja Feldmann, Wolfgang Kellerer, Aiko Pras, Henning Schulzrinne, Georgios Smaragdakis, Matthias Wählisch, and Klaus Wehrle. The Dagstuhl beginners guide to re-

producibility for experimental networking research. *ACM SIGCOMM Computer Communication Review*, 49(1):24–30, 2019.

[12] Andy Bavier, Mark Berman, Marshall Brinn, Rick McGeer, Larry Peterson, and Glenn Ricart. Realizing the global edge cloud. *IEEE Communications Magazine*, 56(5):170–176, 2018.

[13] Ignacio Bermudez, Stefano Traverso, Marco Mellia, and Maurizio Munafò. Exploring the cloud from passive measurements: The amazon AWS case. In *IEEE INFOCOM*, pages 230–234, 2013.

[14] M. Bernaschi, F. Cacace, A. Pescape, and S. Za. Analysis and experimentation over heterogeneous wireless networks. In *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, pages 182–191, Feb 2005.

[15] Manuele Bonaccorsi, Laura Fiorini, Filippo Cavallo, Alessandro Saffiotti, and Paolo Dario. A cloud robotics solution to improve social assistive robots for active and healthy aging. *International Journal of Social Robotics*, 8(3):393–408, 2016.

[16] Alessio Botta, Salvatore D'Antonio, Antonio Pescapé, and Giorgio Ventre. Bet: A hybrid bandwidth estimation tool. In *ICPADS (2)*, pages 520–524, 2005.

[17] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.

[18] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I. Tsang,

S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl. Reducing internet latency: A survey of techniques and their merits. *IEEE Communications Surveys Tutorials*, 18(3):2149–2196, 2016.

[19] C. Guo et al. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. *ACM SIGCOMM Computer Communication Review*, pages 139–152, 2015.

[20] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. How sensitive are online gamers to network quality? *Communications of the ACM*, 49 (11):34–38, 2006.

[21] David Chou, Tianyin Xu, Kaushik Veeraraghavan, Andrew Newell, Sonia Margulis, Lin Xiao, Pol Mauri Ruiz, Justin Meza, Kiryong Ha, Shruti Padmanabha, et al. Taiji: managing global user traffic for large-scale internet services at the edge. In *ACM SOSP'19*, pages 430–446, 2019.

[22] S. Choy, B. Wong, G. Simon, and C. Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *IEEE/ACM NetGames*, pages 1–6, 2012.

[23] Cisco Annual Internet Report (2018–2023). `https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html\#~mobile-forecast`.

[24] Cloud Performance Benchmark. Technical report, ThousandEyes, 2019.

[25] Cloud-to-user latency dataset collected by Traffic Research Group. `http://traffic.comics.unina.it/cloud`.

[26] Lorenzo Corneo and Per Gunningberg. Scheduling at the edge for assisting cloud real-time systems. In *Proceedings of the 2018 workshop on theory and practice for integrated cloud, fog and edge computing paradigms*, pages 9–14, 2018.

[27] Bin Dai, Guan Xu, Bengxiong Huang, Peng Qin, and Yang Xu. Enabling network innovation in data center networks with software defined networking: A survey. *Journal of Network and Computer Applications*, 94:33–49, 2017.

[28] Dataset collecting Available bandwidth and achievable throughput measurements on Mobile Broadband networks. `https://doi.org/10.5281/zenodo.1300512`.

[29] Docker image for experiments on MONROE platform. `https://hub.docker.com/r/fabpal92/abw_ach_sometime`.

[30] Docker website. `https://www.docker.com`.

[31] D. Emma, A. Pescape, and G. Ventre. Analysis and experimentation of an open distributed platform for synthetic traffic generation. In *Proceedings. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004.*, pages 277–283, May 2004.

[32] D. Emma, A. Pescapè, and G. Ventre. Analysis and experimentation of an open distributed platform for synthetic traffic generation. *Proceedings of FTDCS*, 2004. doi: 10.1109/ftdcs.2004.1316627.

[33] Ericsson Mobility Report for 2020. https://www.ericsson.com/en/mobility-report/dataforecasts.

[34] Arsham Farshad, Myungjin Lee, Mahesh K. Marina, and Francisco Garcia. On the impact of 802.11n frame aggregation on end-to-end available bandwidth estimation. *2014 11th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2014*, pages 108–116, 2014.

[35] J. L. Garcia-Dorado and S. G. Rao. Cost-aware multi data-center bulk transfers in the cloud from a customer-side perspective. *IEEE Trans. Cloud Comput.*, 7(1):34–47, Jan 2019. ISSN 2372-0018.

[36] GARR network infrastructure backbones. https://www.garr.it/en/infrastructures/network-infrastructure/backbones.

[37] Gartner Says Worldwide IaaS Public Cloud Services Market Grew 37.3% in 2019. https://www.gartner.com/en/newsroom/press-releases/2020-08-10-gartner-says-worldwide-iaas-public-cloud-services-market-grew-37-point-3-percent-in-2019.

[38] Joseph L Gastwirth, Yulia R Gel, and Weiwen Miao. The impact of Levene's test of equality of variances on statistical theory and practice. *Statistical Science*, pages 343–360, 2009.

[39] Mihaela Gheorghe. Mobile cloud computing for telemedicine solutions. *Informatica Economica*, 18(4), 2014.

[40] Tuan Nguyen Gia, Mingzhe Jiang, Amir-Mohammad Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*, pages 356–363. IEEE, 2015.

[41] E. Goldoni, G. Rossi, and A. Torelli. Assolo, a new method for available bandwidth estimation. In *2009 Fourth International Conference on Internet Monitoring and Protection*, pages 130–136, May 2009.

[42] Emanuele Goldoni and Marco Schivi. *End-to-End Available Bandwidth Estimation Tools, An Experimental Comparison*, pages 171–182. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-12365-8.

[43] Lena Griebel, Hans-Ulrich Prokosch, Felix Köpcke, Dennis Toddenroth, Jan Christoph, Ines Leb, Igor Engel, and Martin Sedlmayr. A scoping review of cloud computing in healthcare. *BMC medical informatics and decision making*, 15(1):1–16, 2015.

[44] David Griffin, Truong Khoa Phan, Elisa Maini, Miguel Rio, and Pieter Simoens. On the feasibility of using current data centre infrastructure for latency-sensitive applications. *IEEE Trans. Cloud Comput.*, 2018.

[45] Cesar D. Guerrero and Miguel A. Labrador. Traceband: A fast, low overhead and accurate tool for available bandwidth estimation and mon-

itoring. *Computer Networks*, 54(6):977 – 990, 2010. ISSN 1389-1286. New Network Paradigms.

[46] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. *ACM SIGCOMM Computer Communication Review*, 45(4):139–152, 2015.

[47] P. Ha and L. Xu. Available bandwidth estimation in public clouds. In *IEEE INFOCOM WKSHPS*, pages 238–243, 2018.

[48] Waqar Haider, Waheed Iqbal, Fawaz S. Bokhari, and Faisal Bukhari. On providing response time guarantees to a cloud-hosted telemedicine web service. In Yin Zhang, Limei Peng, and Chan-Hyun Youn, editors, *Cloud Computing*, pages 234–243, Cham, 2016. Springer International Publishing. ISBN 978-3-319-38904-2.

[49] Akram Hakiri, Bassem Sellami, Prithviraj Patil, Pascal Berthou, and Aniruddha Gokhale. Managing Wireless Fog Networks using Software-Defined Networking. 2017.

[50] Peter W Hamilton, Yinhai Wang, and Stephen J McCullough. Virtual microscopy and digital pathology in training and education. *Apmis*, 120 (4):305–315, 2012.

[51] Cheol-Ho Hong and Blesson Varghese. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys (CSUR)*, 52(5):1–37, 2019.

[52] Ningning Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, Aug 2003. ISSN 0733-8716.

[53] Junxian Huang, Feng Quian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. *ACM SIGCOMM Conference*, pages 363–374, 2013. ISSN 0146-4833.

[54] Md Mofijul Islam, Md Abdur Razzaque, Mohammad Mehedi Hassan, Walaa Nagy Ismail, and Biao Song. Mobile cloud-based big healthcare data processing in smart cities. *IEEE Access*, 5:11887–11899, 2017.

[55] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, 11(4):537–549, August 2003. ISSN 1063-6692.

[56] Yuchen Jin, Sundararajan Renganathan, Ganesh Ananthanarayanan, Junchen Jiang, Venkata N Padmanabhan, Manuel Schroder, Matt Calder, and Arvind Krishnamurthy. Zooming in on wide-area latencies to a global cloud provider. In *ACM SIGCOMM*, pages 104–116, 2019.

[57] Z. Jin and Y. Chen. Telemedicine in the cloud era: Prospects and challenges. *IEEE Pervasive Computing*, 14(1):54–61, 2015.

[58] B. Karacali, J. M. Tracey, P. G. Crumley, and C. Basso. Assessing

cloud network performance. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, 2018.

[59] Roger P Karrer, Istvan Matyasovszki, Alessio Botta, and Antonio Pescapè. Experimental evaluation and characterization of the magnets wireless backbone. In *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, pages 26–33. ACM, 2006.

[60] Roger P Karrer, Istvan Matyasovszki, Alessio Botta, and Antonio Pescapè. Magnets-experiences from deploying a joint research-operational next-generation wireless access network testbed. In *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom*. IEEE, 2007.

[61] Kentik Report: It's a Multi-cloud, Cost-containment World. `https://www.kentik.com/blog/report-multi-cloud-cost-containment-world/`.

[62] D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[63] A. Ksentini, P. A. Frangoudis, P. C. Amogh, and D. Nikaein. Providing low latency guarantees for slicing-ready 5G systems via two-level MAC scheduling. *IEEE Network*, 32(6):116–123, 2018.

[64] James F Kurose. *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005.

[65] M. Kwon, Z. Dou, W. Heinzelman, T. Soyata, H. Ba, and J. Shi. Use of network latency profiling and redundancy for cloud server selection. In *IEEE CLOUD*, pages 826–832, 2014.

[66] Asif Ali Laghari, Hui He, Muhammad Shafiq, and Asiya Khan. Assessing effect of cloud distance on end user's Quality of Experience (QoE). In *IEEE ICCC'16*, pages 500–505, 2016.

[67] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *ACM IMC*, pages 1–14, 2010.

[68] Mingzhe Li, M. Claypool, and R. Kinicki. Wbest: A bandwidth estimation tool for ieee 802.11 wireless networks. In *2008 33rd IEEE Conference on Local Computer Networks (LCN)*, pages 374–381, Oct 2008.

[69] Yuanjie Li, Haotian Deng, Chunyi Peng, Zengwen Yuan, Guan-Hua Tu, Jiayao Li, and Songwu Li. icellular: Device-customized cellular network access on commodity smartphones. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 643–656, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-29-4.

[70] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Mark Badger, and Dawn Leaf. Nist cloud computing reference architecture, 2011-09-08 2011.

[71] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner.

Openflow: Enabling innovation in campus networks. *SIGCOMM Computer Communnication Review*, 38(2):69–74, March 2008.

[72] Péter Megyesi, Alessio Botta, Giuseppe Aceto, Antonio Pescapè, and Sándor Molnár. Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, 2016. ISSN 0140-3664.

[73] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, USA, 2011.

[74] M. Menzel and R. Ranjan. CloudGenius: decision support for web server cloud migration. In *Proceedings of the 21st International Conference on World Wide Web*, pages 979–988, 2012.

[75] Dragorad Milovanovic and Zoran Bojkovic. Cloud-based iot healthcare applications: Requirements and recommendations. *International Journal of Internet of Things and Web Services*, 2:60–65, 2017.

[76] Modified version of Yaz for Mobile Broadband networks. `http://traffic.comics.unina.it/sdn/yaz_modified.tar.gz`.

[77] J. C. Mogul and L. Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012.

[78] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. Pruning edge research with latency shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, pages 182–189, 2020.

[79] P. Mulinka and L. Kencl. Learning from Cloud latency measurements. In *IEEE ICCW*, pages 1895–1901, 2015.

[80] P. Mulinka, P. Casas, and L. Kencl. Hi-Clust: Unsupervised analysis of cloud latency measurements through hierarchical clustering. In *IEEE CloudNet*, pages 1–7, 2018.

[81] Seong K Mun, Al M Elsayed, Walid G Tohme, and Y Chris Wu. Teleradiology/telepathology requirements and implementation. *Journal of medical systems*, 19(2):153–164, 1995.

[82] M. Murray, S. Smallen, O. Khalili, and M. Swany. Comparison of end-to-end bandwidth measurement tools on the 10gige teragrid backbone. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, GRID '05, pages 300–303, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9492-5.

[83] P. Nanda and R. C. Fernandes. Quality of service in telemedicine. In *First International Conference on the Digital Society (ICDS'07)*, 2007.

[84] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. A first look at commercial 5g performance on smartphones. In *Proceedings of The Web Conference 2020*, WWW '20, pages 894–905, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233.

[85] Takashi Oshiba, Kousuke Nogami, Koichi Nihei, and Kozo Satoda. Robust available bandwidth estimation against dynamic behavior of packet scheduler in operational LTE networks. *Proceedings - IEEE Symposium*

on Computers and Communications, 2016-Augus:1276–1283, 2016. ISSN 15301346.

[86] Fabio Palumbo, Giuseppe Aceto, Alessio Botta, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. Characterization and analysis of cloud-to-user latency: The case of azure and aws. *Computer Networks*, 184, 2021. ISSN 1389-1286.

[87] Anup Kumar Paul, Atsuo Tachibana, and Teruyuki Hasegawa. Next-fit: Available bandwidth measurement over 4g/lte networks – a curve-fitting approach. *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 25–32, 2016.

[88] Utpal Paul, Anand Prabhu Subramanian, Milind Madhav Buddhikot, and Samir R Das. Understanding traffic dynamics in cellular data networks. In *2011 Proceedings IEEE INFOCOM*, pages 882–890. IEEE, 2011.

[89] Vern Paxson. Strategies for sound internet measurement. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, pages 263–271, New York, NY, USA, 2004. ACM. ISBN 1-58113-821-0.

[90] V. Persico, P. Marchetta, A. Botta, and A. Pescapé. Measuring network throughput in the cloud: The case of Amazon EC2. *Computer Networks*, 93:408–422, 2015.

[91] V. Persico, P. Marchetta, A. Botta, and A. Pescapé. On network throughput variability in Microsoft Azure cloud. In *IEEE GLOBECOM'15*, pages 1–6, 2015.

[92] V. Persico, A. Montieri, and A. Pescapé. On the network performance of Amazon S3 cloud-storage service. In *IEEE Cloudnet*, pages 113–118, 2016.

[93] Valerio Persico, Alessio Botta, Pietro Marchetta, Antonio Montieri, and Antonio Pescapé. On the performance of the wide-area networks interconnecting public-cloud datacenters around the globe. *Computer Networks*, 112:67 – 83, 2017.

[94] Valerio Persico, Alessio Botta, Pietro Marchetta, Antonio Montieri, and Antonio Pescapé. On the performance of the wide-area networks interconnecting public-cloud datacenters around the globe. *Computer Networks*, 112:67–83, 2017.

[95] Diana Andreea Popescu and Andrew W Moore. PTPmesh: Data center network latency measurements using PTP. In *IEEE MASCOTS'17*, pages 73–79, 2017.

[96] Diana Andreea Popescu and Andrew W Moore. A first look at data center network condition through the eyes of PTPmesh. In *IEEE TMA'18*, pages 1–8, 2018.

[97] Diana Andreea Popescu, Noa Zilberman, and Andrew William Moore. Characterizing the impact of network latency on cloud-based applications' performance. Technical Report UCAM-CL-TR-914, Univ. of Cambridge, 2017.

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

[99] Costin Raiciu, Mihail Ionescu, and Dragos Niculescu. Opening up black box networks with CloudTalk. In *ACM HotCloud*, page 6, 2012.

[100] Daniel Raumer, Lukas Schwaighofer, and Georg Carle. Monsamp: A distributed sdn application for qos monitoring. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 961–968. IEEE, 2014.

[101] Vinay Joseph Ribeiro, Rudolf H Riedi, Richard G Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and active measurement workshop*, 2003.

[102] Alessio Sacco, Flavio Esposito, Guido Marchetto, Grant Kolar, and Kate Schwetye. On edge computing for remote pathology consultations and computations. *IEEE Journal of Biomedical and Health Informatics*, 24(9):2523–2534, 2020. doi: 10.1109/JBHI.2020.3007661.

[103] Natsuhiko Sato, Takashi Oshiba, Kousuke Nogami, Anan Sawabe, and Kozo Satoda. Experimental comparison of machine learning-based available bandwidth estimation methods over operational LTE networks. *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 339–346, 2017. ISSN 15301346.

[104] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, and Nigel Davies. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009. ISSN 15361268.

[105] Shahab Shamshirband, Mahdis Fathi, Anthony T. Chronopoulos, Antonio Montieri, Fabio Palumbo, and Antonio Pescapé. Computational

intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues. *Journal of Information Security and Applications*, 55:102582, 2020. ISSN 2214-2126.

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

[107] Y. Shi, G. Ding, H. Wang, H. E. Roman, and S. Lu. The fog computing service for healthcare. In *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech)*, pages 1–5, 2015.

[108] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and kc claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *Proceedings of the 6th International Conference on Passive and Active Network Measurement*, PAM'05, pages 306–320, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-25520-6, 978-3-540-25520-8.

[109] Joel Sommers, Paul Barford, and Walter Willinger. Laboratory-based calibration of available bandwidth estimation tools. *Microprocessors and Microsystems*, 31(4):222 – 235, 2007. ISSN 0141-9331. Special Issue with selected papers from the 11th {IEEE} Symposium on Computers and Communications (ISCC'06).

[110] Lixing Song and Aaron Striegel. Leveraging Frame Aggregation for Estimating WiFi Available Bandwidth. *2017 14th Annual IEEE In-*

ternational Conference on Sensing, Communication, and Networking, SECON 2017, (March), 2017.

[111] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 39–44, New York, NY, USA, 2003. ACM. ISBN 1-58113-773-7.

[112] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 39–44, New York, NY, USA, 2003. ACM. ISBN 1-58113-773-7.

[113] L. A. Tawalbeh and S. Habeeb. An integrated cloud based healthcare system. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 268–273, 2018.

[114] Bernard Têtu, Guy Paré, Marie-Claude Trudel, Julien Meyer, Peter V Gould, Stephan Saikali, Michèle Orain, Lyne Nadeau, and Bich N Nguyen. Whole-slide imaging-based telepathology in geographically dispersed healthcare networks. the eastern québec telepathology project. *Diagnostic Histopathology*, 20(12):462–469, 2014.

[115] The MONROE project website. `https://www.monroe-project.eu`.

[116] O. Tomanek, P. Mulinka, and L. Kencl. Multidimensional cloud latency monitoring and evaluation. *Computer Networks*, 107(Part 1):104–120, 2016.

[117] Radu Tudoran, Alexandru Costan, Gabriel Antoniu, and Luc Bougé. A performance evaluation of Azure and Nimbus clouds for scientific applications. In *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, pages 1–6, 2012.

[118] V. Uhlir, O. Tomanek, and L. Kencl. Latency-based benchmarking of cloud service providers. In *IEEE/ACM UCC*, pages 263–268, 2016.

[119] Niels LM Van Adrichem, Christian Doerr, and Fernando A Kuipers. Opennetmon: Network monitoring in openflow software-defined networks. In *NOMS, 2014 IEEE*, pages 1–8. IEEE, 2014.

[120] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 29–36, 2012.

[121] Lidong Wang and Cheryl Ann Alexander. Telemedicine based on mobile devices and mobile cloud computing. *International Journal of Cloud Computing and Services Science*, 3(1):26, 2014.

[122] Y. A. Wang, C. Huang, J. Li, and K. W. Ross. Estimating the performance of hypothetical cloud service deployments: A measurement-based approach. In *IEEE INFOCOM*, pages 2372–2380, 2011.

[123] Ronald S Weinstein, Michael R Descour, Chen Liang, Achyut K Bhattacharyya, Anna R Graham, John R Davis, Katherine M Scott, Lynne Richter, Elizabeth A Krupinski, Janusz Szymus, et al. Telepathology overview: from concept to implementation. *Human pathology*, 32(12): 1283–1299, 2001.

[124] What is Mobile Cloud Computing? By IBM. `https://www.ibm.com/blogs/cloud-computing/2013/06/25/mobile-cloud-computing`.

[125] Frank Wilcoxon, SK Katti, and Roberta A Wilcox. Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. *Selected tables in mathematical statistics*, 1:171–259, 1970.

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

[127] Zhe Wu and Harsha V Madhyastha. Understanding the latency benefits of multi-cloud webservice deployments. *ACM SIGCOMM Computer Communication Review*, 43(2):13–20, 2013.

[128] Xiufeng Xie, Xinyu Zhang, and Shilin Zhu. Accelerating mobile web loading using cellular link information. In *Proceedings of MobySys '17*, MobiSys '17, pages 427–439, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4928-4.

[129] D. Xu and D. Qian. A bandwidth adaptive method for estimating end-to-end available bandwidth. In *2008 11th IEEE Singapore International Conference on Communication Systems*, pages 543–548, Nov 2008.

[130] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All one needs to know about fog computing and related edge computing

paradigms: A complete survey. *Journal of Systems Architecture*, 98: 289–330, 2019. ISSN 1383-7621.

[131] Curtis Yu, Cristian Lumezanu, Yueping Zhang, Vishal Singh, Guofei Jiang, and Harsha V Madhyastha. Flowsense: Monitoring network utilization with zero measurement cost. In *PAM*, pages 31–41. Springer, 2013.

[132] Z. Hu et al. The need for end-to-end evaluation of cloud availability. In *International Conference on Passive and Active Network Measurement*, pages 119–130, 2014.

[133] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y Zhao, et al. Packet-level telemetry in large datacenter networks. *ACM SIGCOMM Computer Communication Review*, 45(4):479–491, 2015.