Politecnico di Milano

Dipartmento di Elettronica, Informazione e Bioingegneria

Master of Science in Telecommunication Engineering

Techno-Economic Analysis of Mobile Edge Computing for the Base 5G project

Supervisor:

Prof. Giacomo Verticale

Candidate:

Ahanonu Chinwendu Jane

Matricola: 925502

ACADEMIC YEAR 2020-2021

ACKNOWLEDGEMENT

A huge thanks to my thesis supervisor, Professor Giacomo Verticale, for his unwavering patience, understanding and immense support while working on this thesis. I am extremely fortunate and honoured to have worked with him as my supervisor. His consistent help and feedbacks have proven to be instrumental in the completion of my Master's degree, thereby achieving my goals.

ABSTRACT

The 5G network is vastly becoming available in many countries, including Italy, and is emerging as the new reference architecture for the global mobile and fixed telecommunication network. 5G is not only an evolution of 4G in terms of performance, but it also creates a breaking point with respect to previous generations: 5G will support a number of diversified vertical sectors, targeting different types of users and services.

This thesis work is based on the reference architecture of the Base5G project for techno-economic analysis. Its aim is to experiment how 5G can help value IoT applications, while exploring the new technology called “Network slicing”. It then presents Base5G mathematical models for identifying different trade-offs in the deployment of MEC in a real operator network.

Mobile Edge Computing (MEC) is a network architecture concept that enables the execution of the various network services at the edge of the cellular network allowing the mobile users to benefit from a lower latency and network congestion since the services are delivered from a geographical area that is very close to the user location. To be effective, MEC requires that mobile operators open their Radio Access Network (RAN) to authorized service operators and content providers.

The task of this thesis work is to analyse and determine the appropriate positions to activate MEC for optimal coverage of several base station sites. We have data from a mobile operator about the network latency in the backhauling and in the core part of the network and want to evaluate the cost in terms of the number of MEC-enabled sites in order to provide different real-time applications. By leveraging delay measurements collected on field, the Base5G models make it possible for an operator to obtain a rough estimate of the cost necessary to enable Ultra Reliable Low Latency (URLL) services for its customers. These models also

make it possible to identify the optimal locations of the mobile radio sites that should be upgraded to support MEC in order to maximize the speed of deployment of URLL services. Numerical results applied to measurements collected show that, without MEC, URLL services can be deployed only to a small part of the network subscribers. On the other hand, if MEC could be enabled in all the candidate sites, then URLL services could be offered to almost all the network subscribers

ABSTRACT (ITALIANO)

La rete 5G sta diventando ampiamente disponibile in molti paesi, Italia compresa, e si sta affermando come la nuova architettura di riferimento per la rete globale di telecomunicazioni mobili e fisse. Il 5G non è solo un'evoluzione del 4G in termini di prestazioni, ma crea anche un punto di rottura rispetto alle generazioni precedenti: il 5G supporterà una serie di settori verticali diversificati, rivolgendosi a diverse tipologie di utenti e servizi.

Questo lavoro di tesi si basa sull'architettura di riferimento del progetto Base5G per l'analisi tecnico-economica. Il suo obiettivo è sperimentare come il 5G può aiutare a valorizzare le applicazioni IoT, esplorando la nuova tecnologia chiamata "Network slicing". Presenta quindi modelli matematici Base5G per identificare diversi compromessi nell'implementazione di MEC in una rete di operatori reali.

Mobile Edge Computing (MEC) è un concetto di architettura di rete che consente l'esecuzione dei vari servizi di rete ai margini della rete cellulare consentendo agli utenti mobili di beneficiare di una minore latenza e congestione della rete poiché i servizi sono forniti da un'area geografica che è molto vicino alla posizione dell'utente. Per essere efficace, MEC richiede che gli operatori mobili aprano la loro rete di accesso radio (RAN) agli operatori di servizi autorizzati e ai fornitori di contenuti.

Il compito di questo lavoro di tesi è analizzare e determinare le posizioni appropriate per attivare MEC per una copertura ottimale di diversi siti di stazioni base. Abbiamo dati da un operatore mobile sulla latenza della rete nel backhauling e nella parte centrale della rete e vogliamo valutare il costo in termini di numero di siti abilitati MEC per fornire diverse applicazioni in tempo reale. Sfruttando le misurazioni del ritardo raccolte sul campo, i modelli Base5G consentono a un operatore di ottenere una stima approssimativa del costo necessario per abilitare i servizi URLL (Ultra Reliable Low Latency) per i propri clienti. Questi modelli consentono inoltre di identificare le posizioni ottimali dei siti radio mobili che

dovrebbero essere aggiornati per supportare MEC al fine di massimizzare la velocità di implementazione dei servizi URLL.

I risultati numerici applicati alle misurazioni raccolte mostrano che, senza MEC, i servizi URLL possono essere distribuiti solo a una piccola parte degli abbonati alla rete. Se invece MEC fosse abilitato in tutti i siti candidati, i servizi URLL potrebbero essere offerti a quasi tutti gli abbonati alla rete

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

1 Introduction

With the rapid development of mobile communications and the explosive usage of mobile devices (i.e., smart phones, laptops, tablets, etc.), the mobile Internet facilitates us with a pervasive and powerful platform to provide emerging applications. However, many mobile devices usually have limited computation capabilities and battery power. Migrating computational tasks from the distributed devices to the infrastructure-based cloud servers has the potential to address the aforementioned issues. Mobile Edge Computing (MEC) is an emerging paradigm, which aims to provide better services by moving infrastructure-based cloud resources (computation, storage, bandwidth, etc.) to the edge of the network. Different from the traditional cloud, MEC is close to the mobile users. This reduces the access delay and the cost of using the cloud service. MEC is rapidly becoming a key technology of 5G and beyond, achieving the key technical indicators of 5G business, such as ultra-low latency, ultra-high energy efficiency, and ultra-high reliability [1].

5G is designed to be a multi-service network supporting multiple verticals with a diverse set of performance requirements. The key to realize this vision is slicing the physical network into multiple isolated logical networks on a per-service basis. Network slicing is the technical mean for allowing the coexistence of different verticals over the same infrastructure. According to the Next Generation Mobile Networks (NGMN) Alliance, a network slice is a set of network functions, and the resources to run these functions, forming a logical network that meets the requirements of a given service[6]. Network slicing, which virtualizes both the radio access and core networks, will enable per-service performance levels and isolation. It will support a model in which the mobile service operator operates as a resource broker,

which pools resources from different infrastructure operators and provides network slices. Each slice is then operated by a tenant, which offers the service to the end-user.

Network slicing needs to be enhanced with Mobile Edge Computing (MEC). In the context of 5G network slicing, MEC is a valuable tool to efficiently guarantee the delay requirements of URLLC services. However, to practically employ this technology in conjunction with network slicing, cellular operators need to provide a standardized access point between the MEC and network slicing architecture that can enable a paradigm of "service as a slice". This goal is achieved by interfacing the MEC architecture with the core network user plane functions (UPF) that are active for each individual network slice [2]

1.1 Thesis Aim

- To experiment how 5G MEC can help value IOT applications.
- Technical-economic analysis of slicing (a new technology in 5G). Evaluation of how much we must spend in order to provide a low latency in Italy

1.2 Objectives

The goal of this work is to do a technical-economic analysis of Mobile Edge Computing using acquired data from Vodafone to evaluate the cost in terms of the number of MEC-enabled sites in order to provide different real-time applications.

There are 4 Scenarios to be considered for analysis:

- All the gateways have the data centre (in this case, the delay is the access delay)
- Only gateway that has data centre is Milan (MI01)
- MI01 + one additional gateway
- Milan + any number of other gateways

1.3 Thesis Outline

This thesis is organized in the following way:

Chapter 1 (Introduction) briefly introduces us into the concept of mobile edge computing, network slicing and its value in the mobile operator networks. The aims and objective of this thesis work is also presented chapter.

Chapter 2 (Background and Related Surveys) describes an overview of 5G network slicing and 5G MEC that mainly encompass; definition, architecture, application in the core network and radio access network of 5G, mobile edge computing advantages, and related surveys that are presented recently. Most importantly, the related concepts and technologies appear in this chapter

Chapter 3 (Use case) illustrates edge computing applications and use cases, and then provides a description of the Base5G reference architecture.

Chapter 4 (Mathematical model) presents the mathematical model for techno-economic analysis of MEC deployment.

Chapter 5 (Results Discussion) provides preliminary techno-economic analysis obtained by applying the Base5G models to performance data obtained from measurements on field. Some of the possible future works are also stated

Chapter 6 (Conclusions) Finally, this chapter comes up with the conclusion of this project.

CHAPTER TWO

2 Background and Related Surveys

2.1 Overview of 5G Network Slicing

Upcoming 5G networks are designed to support a plethora of applications that are characterized by heterogeneous service requirements. Such services are clustered in 3 main categories defined as [3]:

- enhanced multi-broadband (eMBB) services: applications that require high data-rate (e.g., 4K streaming)
- ultra-reliable low latency communication (uRLLC): applications that require a communication characterized by low decoding errors and delay (e.g., mission-critical applications, vehicular communications)
- massive machine-type communication (mMTC): applications run by energy-limited devices that are employed for monitoring and sensing purposes (e.g., IoT applications for smart cities)

Every category is identified by specific requirements that can be hardly accommodated by a single network infrastructure. To efficiently answer these unprecedented market needs, the "one-size-fits-all" network architectural approach currently employed in traditional networks is no longer suitable to simultaneously fulfil contrasting QoS requirements [4]. For this reason, 5G networks leverage the recent advances in network function virtualization (NFV) and software-defined networking (SDN) to tailor the network architecture according to each specific user application. In details, the network is "sliced" in different logical networks that share a common pool of resources provided by the physical network infrastructure. The flexibility provided by this approach, denoted as network slicing, allows each network

operator to optimize and dynamically structure its own network based on the provisioned service [5]. In addition to the performance gains in terms of service delivery efficiency, this technology also improves the economic revenue of the various service providers thanks to the shared physical network infrastructure [6]. In Fig. 1, an example of end-to-end network slicing is shown.

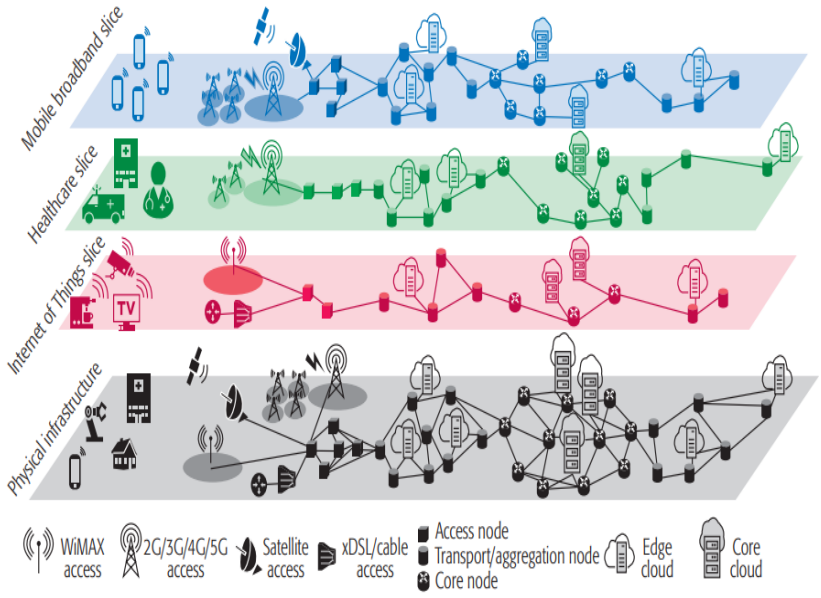

Figure 1 5G End-to-end network slicing [5]

As highlighted by the figure, the network slicing concept can be implemented at different networks levels, that are identified as core network, transport network and radio access network. The main difference characterizing the network slice architecture on the various network levels is the physical resource that is shared between the virtual operators. Specifically, network slices that are executed on the core and transports networks usually share computational resources such as CPUs to sustain the service provisioning connectivity, whereas network slices that are active on the radio interface share the radio spectrum to independently multiplex the related applications among multiple mobile users.

2.2 Network slicing architecture

The network slicing architecture can be generally represented as composed by three main layers: the service layer, the network function layer, and the infrastructure layer. A representation of such architecture is shown in Fig. 2.

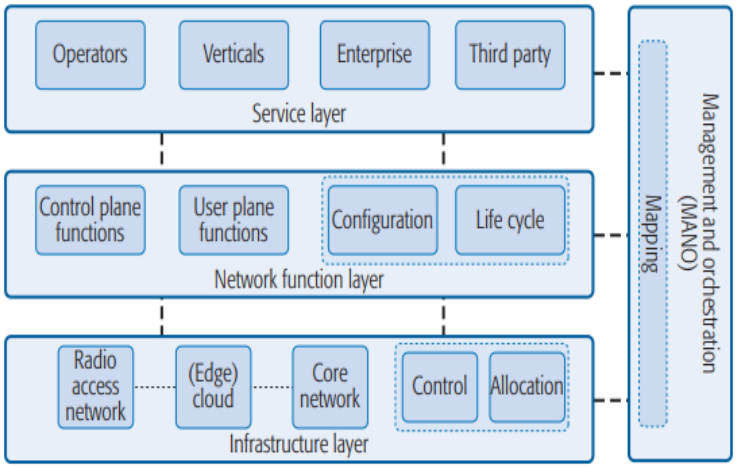

Figure 2 General 5G network slicing architecture [4]

In detail, the tasks of each layer can be summarized as follows [4]:

- Service layer: it represents the user application upon which a specific network slice is built. Each user application is formally defined as service instance that can be either provided by the virtual network operator or by a third-party business entity. The fulfillment of the application QoS is regulated by a set of service level agreements (SLA) that must be guarantee by a suitable network slice.
- Network function layer: it implements the mapping procedure that translates the service requirements of each service instance into specific network characteristics that are required to effectively run the application. This process is performed by the virtual network operator (i.e. the network slice owner) that employs a set of network slice templates, denoted as network slice blueprints, to quantitatively formalize the SLA requirements. Every network slice blueprint consists of a complete description of the

structure as well as configuration for how to instantiate and control the network slice during its life cycle. Note that the same network slice instance may be shared by multiple service instances providing similar network applications. The network slice blueprint is then used to implement the actual logical network by means of a set of network functions.

- Infrastructure layer: it represents the common pool of resources that is used by the network functions to run the related network slice. The resource management among multiple network slices is performed by the network function virtualization orchestrator. The latter allocates a suitable number of physical resources to each network function to ensure the accommodation of the SLA requirements requested by each service instance.

The functionality of the above layers is explicitly monitored and managed by the network management and orchestration (MANO) plane. The latter oversees the coordination between the different layer tasks as well as the monitoring of the full life cycle of each network slice. Specifically, the activities of the MANO controller are [5]:

1. Creation of virtual network instances upon the physical network by using the resources available in the resource layer.
2. Implementation of virtualized network instances by chaining the available network functions available within the network slice instance layer.
3. Maintaining communication between service layer and the network slicing framework to manage the lifecycle of virtual network instances and dynamically adapt or scale the virtualized resources according to the changing context.

2.2.1 Network slicing in the Core Network (CN)

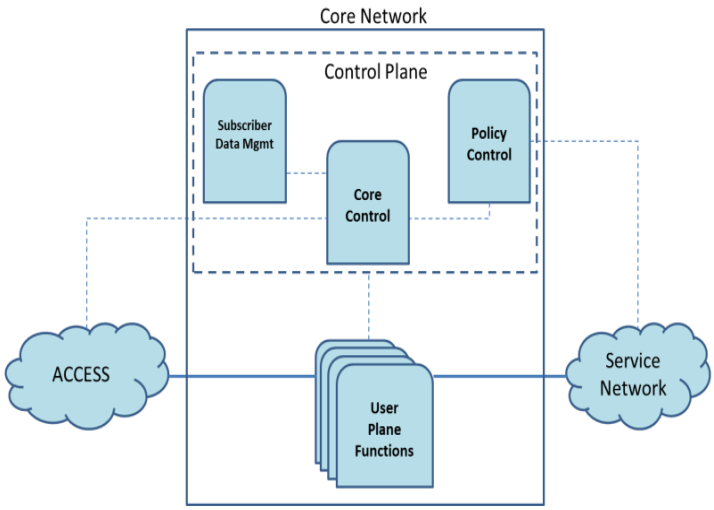

Figure 3 CN control plane and user plane split [7]

Network slicing applied in the 5G core network allows a full separation of the control plane and user plane as shown in Fig. 3 [7]. The advantage of this design choice is the enhancement of the service provisioning flexibility, since the control plane can be considered as a single centralized entity that is agnostic of the activity of the multiple user plane functions. Specifically, the control plane role is limited to user authentication and policy control procedures that are shared by different network slices. Differently, the user plane functionality is fully virtualized and split in multiple user plane functions that are tailored to each specific use case. By ensuring the scalability between the control plane and user plane independent one from the other, the location of the core network is no longer constrained to a single geographical location. This additional degree of freedom allows to place the control plane activities in centralized sites to ease the management operations, whereas each user plane function can be brought closer to the end-user which would benefit from a lower communication latency. This alternative core network architecture is particularly important for boosting the performance of delay sensitive services as they represent a wide portion of the 5G applications. Another practical example of this approach is Mobile Edge Computing

(MEC) that is a network architecture designed to increase the computational capabilities of the RAN by placing computational resources close to the radio interface [8].

2.2.2 Network slicing in the Radio Access Network (RAN)

Network slicing on the radio interface is defined as a set of configuration rules that are associated to each network slice to accomplish the related supported network service. In detail, RAN slicing is implemented by following the design paradigm of "Slice as a service", that consists of a one-to-one mapping between a specific service provided by a business entity and a network slice [3]. In practice, a virtual operator, that shares the RAN infrastructure with other virtual operators, may propose to the customer (the business entity) a network slice (e.g., URLLC for automotive industry) with configurable characteristics. The slice instance is tailored according to the agreed SLA requirements and successively deployed to effectively provide the service to the network users. Each mobile user is notified of the expected slice behavior that is connected to by means of a slice identifier that defines a set of QoS parameters suited for fulfilling the application requirements [3].

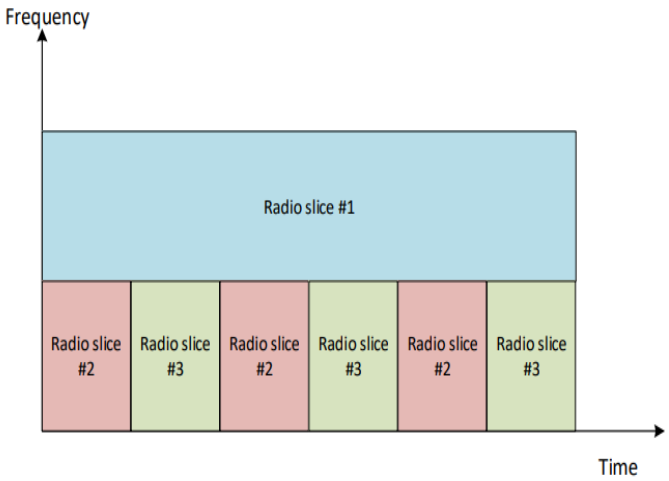

Figure 4 Spectrum allocation in RAN slicing [7]

The radio resource management covers a particularly important role in RAN slicing due to the fact that spectrum resources are limited, and their overprovisioning is not possible. Consequently, the available RAN spectrum is divided in multiple time-frequency slots the are used to simultaneously multiplex different slices as shown in Fig. 4. In this context, two spectrum allocation modes have been proposed, which are denoted as static resource sharing and dynamic resource sharing [7]. The former allows to pre-configure the operational slice bandwidth on a fixed spectrum region, whereas the latter allows a fully reuse of the RAN spectrum by each active network slice. From an optimization performance perspective, a dynamic resource allocation ensures more flexibility to accommodate unexpected traffic demands in each slice by tuning the amount of assigned spectrum resources, however this benefit comes at the cost of a more complex coordination between the various network slices. Differently, a static resource sharing can be easily implemented with limited signaling overhead between different slices, however each slice performance may be hindered by the fixed spectrum allocation that denies possible multiplexing gains derived from under-utilized spectrum resources.

2.3 Enhancing Network Slicing with Mobile Edge Computing (MEC)

5G Mobile Edge Computing (MEC) is a network architecture concept that enables the execution of the various network services at the edge of the cellular network [8]. The core idea is to place computational capabilities employed for the service provisioning close to the base stations within the same RAN. The effect of this alternative network topology allows the mobile users to benefit from a lower latency and network congestion since the processing tasks of the related services are performed in a geographical area that is very close to their location [8]. In the context of 5G network slicing, this technology is a valuable tool to efficiently guarantee the delay requirements of URLLC services. However, to practically

employ this technology in conjunction with network slicing, cellular operators need to provide a standardized access point between the MEC and network slicing architecture that can enable the paradigm of "service as a slice" mentioned earlier. This goal is achieved by interfacing the MEC architecture with the CN user plane functions (UPF) that are active for each individual network slice [8]. Since CN virtualization decouples control plane and user plane functions, network slicing can effectively take advantage of the MEC technology by performing the user plane functions tasks like packet processing and traffic aggregation at the network edge. In this regard, the physical deployment of the UPFs and the MEC resources is chosen by the network operator according to performance and business parameters like sites availability, maintenance costs, service requirements and communication bandwidth availability. It is possible to identify four main physical deployments options that offer different trade-off in terms of operational performance and economic costs. As shown in Fig. 5, the four possible configurations are [8]:

1. MEC and the local UPF collocated with the Base Station.
2. MEC collocated with a transmission node, possibly with a local UPF.
3. MEC and the local UPF collocated with a network aggregation point.
4. MEC collocated with the Core Network functions (i.e., in the same data center).

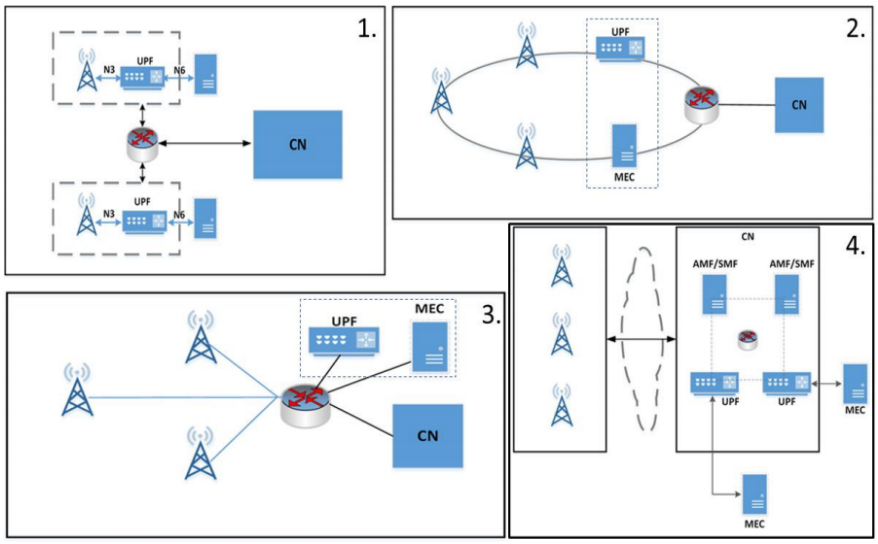

Figure 5 UPF and MEC possible deployments [8]

In general, the closer the UPF and MEC deployments are to the base station the higher is the expected performance gain. However, such solutions usually require more implementations effort that can potentially hinder the operator economic profit.

2.4 Network slicing profit model

Network slicing allows virtual operators to drastically reduce their infrastructure maintenance costs thanks to the shared resource pool [4]. Every network slice can be translated in a set of key performance indicators (KPI) that provides a quantitative indication of the service provisioning quality. Specifically, for every KPI complying with the SLA requirements, the virtual operator gains an economic revenue which also depends on the number of satisfied users (i.e., the slice size). On the other hand, the virtual network functions implementing the network slice require a certain number of heterogeneous resources that are offered by the physical network infrastructure, hence virtual operators incur on some expenditures that depend on the amount of resource consumption (e.g., radio spectrum, power, time, etc.). Note that such costs can be efficiently mitigated by resource allocation policies that achieve the

SLA requirement fulfillment with a limited physical resource consumption [6]. A representation of the discussed network slice profit model is depicted in Fig. 6.

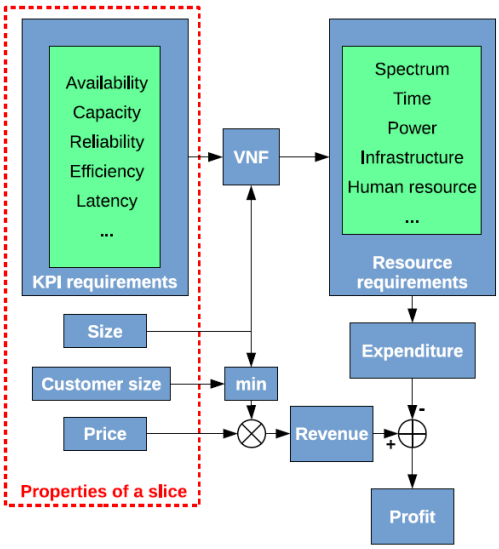

Figure 6 Network slice profit model [6]

2.5 Advantages of Mobile Edge Computing

MEC concept focuses on important metrics, such as delay and high bandwidth that is accomplished by limiting data movement to MEC servers then to centralised servers that has a severe latency cost. Moreover, power consumption is also one of the main concerns. Computational tasks are referred to external resource-rich systems to increase user equipment (UE) battery life. In addition, distributed virtual servers provision scalability and reliability. Some MEC benefits include [9] [10]:

- Mobile network operators could enable RAN access to third party vendors to deploy their applications and services in more flexible and agile manner. These enabling services could generate revenue by charging based on the services used, such as storage, bandwidth, and other IT resources. OTT services and DVR services offered by cable operators may likely be faster since their services could reside in MEC servers.

- Application service providers could gain profit by MEC enabled infrastructure-as-a-service (IaaS) platform at the network edge that make ASPs services scalable along with high bandwidth and low latency. ASPs could also get a real time access to the radio activity that may develop more capable applications. RAN is revamped into Service-Aware RAN (SRAN) that provides information of subscriber location, cell load, network congestion etc.
- End users could experience fast computational applications through offloading technique that is handled by MEC servers within RAN. In addition, tight RAN assimilation and physical close servers could improve user quality of experience (QoE), such as high throughput browsing, video caching, better DNS etc.

2.6 Related Surveys

Table 1. Summary of existing surveys on multi-access edge computing [11]

Theme	Reference	Major Contribution
Architecture and Computation offloading	[12]	- A review on potential MEC architectures and computation ofloading. - A summary of lessons learned from the state-of-the-art research works on computation offloading and a vision for open challenges and future work.
	[13]	- An introduction of MEC and its key enabling technologies, e.g., NFV, SDN, and virtual machines. - A coherent explanation of the MEC reference architecture along with the incentives for the MEC management and orchestration framework.

Resource Allocation	[14]	- An extensive survey of the basic MEC models from the communication perspective, such as computation task models, communication models, computation models of mobile users, and computation models of MEC servers. - A comprehensive review of MEC researches on joint communication and computation resource allocation in three MEC scenarios, single-user, multi-user, and multi-server MEC systems.
	[15] [16]	A complete review of emerging techniques for the convergence and integration of communication, computation, and caching.
Mathematical Frameworks	[17]	A comprehensive survey of research works on making the computation offloading decisions from multiple perspectives.
	[18]	- A fundamental background in game theory (non-cooperative, cooperative, and evolutionary games) and multi-access edge computing. - A review of game theoretical contributions to wireless communication networks and multi-access edge computing and a summary of research directions for theoretical game models and edge computing.
Research Directions	[19]	An overview of MEC background, application use cases, MEC infrastructure, and security and privacy issues.
	[20]	A brief introduction of MEC including concepts, applications, architectures, and open research challenges
	[21] [22]	- A review on how to exploit MEC and other edge computing technologies for the deployment and improvement of various IoT

		applications in the emerging 5G network. - A holistic review of state-of-the-art research works and a discussion on challenges as well as potential future works for MEC-IoT integration.
	[23] [24]	A holistic review and detailed analyses of security and resilience of edge computing technologies, such as fog computing and mobile edge computing

CHAPTER THREE

3 Base 5G Architecture

In this chapter, we shall consider our reference base 5G architecture used for this thesis work analysis. However, we shall first take a detailed look at some of the applications and use cases of mobile edge computing (MEC)

3.1 Applications and Use Cases [25]

There are a number of service scenarios that have been considered within ETSI ISG MEC. This section illustrates various scenarios which can take advantage of Mobile Edge Computing to either increase performance compared to providing such services through the cloud or through core network servers, or to utilize the unique capabilities offered by MEC platforms such as proximity to the user and network edge, serving a highly localized area. It should be noted these examples are non-exhaustive and further service scenarios are available in the ETSI ISG MEC specification for Mobile Edge Computing Service Scenarios, GS MEC 004. Other scenarios which can make use of MEC are also possible.

3.1.1 Augmented Reality

New services become possible when mobile networks supporting high data rates and low latency computation are deployed. One example of such services is Augmented Reality. Augmented reality (AR) is the combination of a view of the real-world environment and supplementary computer-generated sensory input such as sound, video, graphics or GPS data. Augmented reality can enhance the experience of a visitor to a museum or another point of interest. Consider a visitor to a museum, art gallery, city monument, music or sports event, holding their mobile device towards a particular point of interest with the application related

to their visit activated (i.e., the museum application). The camera captures the point of interest and the application displays additional information related to what the visitor is viewing. Augmented reality services require an application to analyse the output from a device's camera and/or a precise location in order to supplement a user's experience when visiting a point of interest by providing additional information to the user about what they are currently experiencing. The application needs to be aware of a user's position and the direction they are facing, either through positioning techniques or through the camera view, or both. After analysing such information, the application can provide additional information in real-time to the user. If the user moves, the information needs to be refreshed. Hosting the Augmented Reality service on a MEC platform instead of in the cloud is advantageous since supplementary information pertaining to a point of interest is highly localised and is often irrelevant beyond the particular point of interest. Figure 7 shows how a MEC platform can be used to provide an Augmented Reality service.

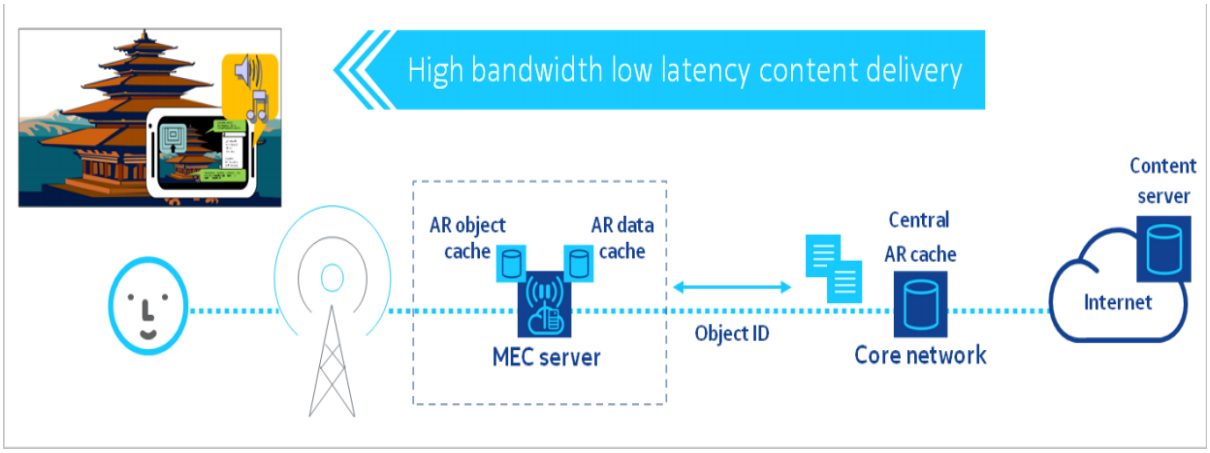

Figure 7 Augmented Reality Service Scenario

3.1.2 Intelligent Video Acceleration

End user Quality of Experience (QoE) and utilization of radio network resources can be improved through intelligent video acceleration. Internet media and file delivery are typically

streamed or downloaded today using Hypertext Transmission Protocol (HTTP) over the TCP protocol. Available capacity can vary by an order of magnitude within seconds (as a result of changes in radio channel conditions, devices entering/leaving network). TCP may not be able to adapt fast enough to rapidlyvarying conditions in the radio access network (RAN). This may lead to under-utilisation of precious radio resources and to a sub-optimal user experience. Figure 8 shows an example of the intelligent video acceleration service scenario which attempts to overcome the challenges described above. In this scenario, a radio analytics application, which resides in a MEC server, provides the video server with an indication on the throughput estimated to be available at the radio downlink interface. This information can be used to assist the TCP congestion control decisions (for example in selecting the initial window size, setting the value of the congestion window during the congestion avoidance phase, and adjusting the size of the congestion window when the conditions on the "radio link" deteriorate). The information can also be used to ensure that the application-level coding matches the estimated capacity at the radio downlink. The video server may use this information to assist TCP congestion control decisions (for example by ensuring that the application level coding matches the estimated capacity at the radio downlink). The content’s time-to-start as well as video-stall occurrences can be reduced, enabling improved video quality and throughput.

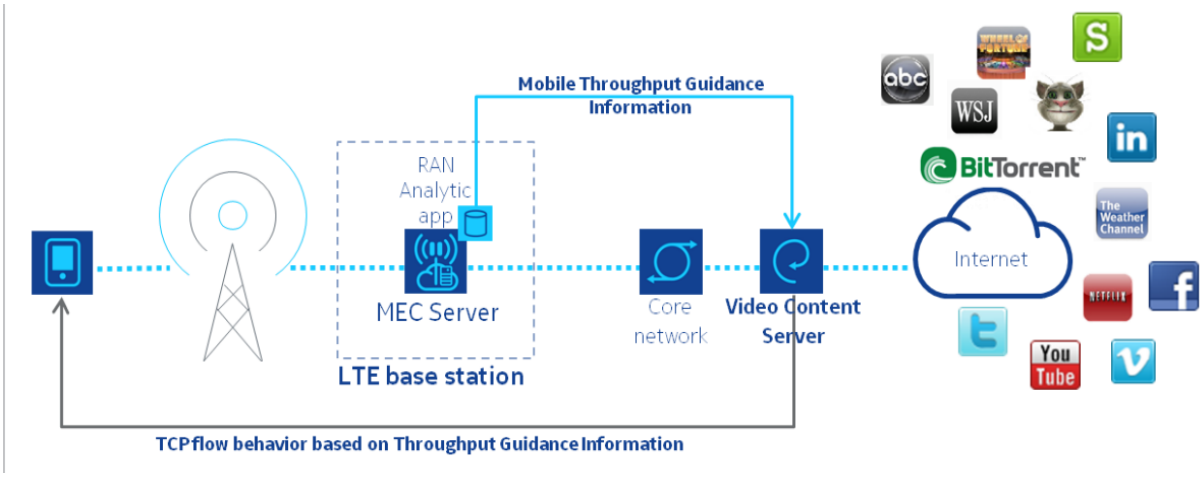

Figure 8 : Intelligent Video Acceleration Service Scenario

3.1.3 Connected Cars

The number of connected vehicles is rapidly growing and will continue to do so over the coming years. Communication of vehicles and roadside sensors with a roadside unit is intended to increase the safety, efficiency, and convenience of the transportation system, by the exchange of critical safety and operational data. The communication can also be used to provide value added services, such as car finder, parking location and to support entertainment services (e.g. video distribution). As the number of connected vehicles increases and use cases evolve, the volume of data will continue to increase along with the need to minimize latency. Whilst data stored and processed centrally may be adequate for some use cases, it can be unreliable and slow for others. LTE can significantly accelerate the deployment of connected vehicle communications. LTE cells can provide "beyond the line of sight" visibility i.e. beyond the range of direct communication between vehicles of 300 – 500m. It can also satisfy the tight latency requirement of connected vehicle communications, below 100ms in some use cases. Messages could be distributed in real time over LTE, eliminating the need to build a countrywide Digital Short-Range Communications (DSRC) network. Cars can leverage their increasingly inbuilt LTE connectivity; in deployments where

DSRC exists, LTE would be able complement it. Mobile Edge Computing can be used to extend the connected car cloud into the highly distributed mobile base station environment, and enable data and applications to be housed close to the vehicles. This can reduce the round trip time of data and enable a layer of abstraction from both the core network and applications provided over the internet. MEC applications can run on MEC servers which are deployed at the LTE base station site to provide the roadside functionality. The MEC applications can receive local messages directly from the applications in the vehicles and the roadside sensors, analyse them and then propagate (with extremely low latency) hazard warnings and other latency-sensitive messages to other cars in the area (as depicted in Figure 9). This enables a nearby car to receive data in a matter of milliseconds, allowing the driver to immediately react.

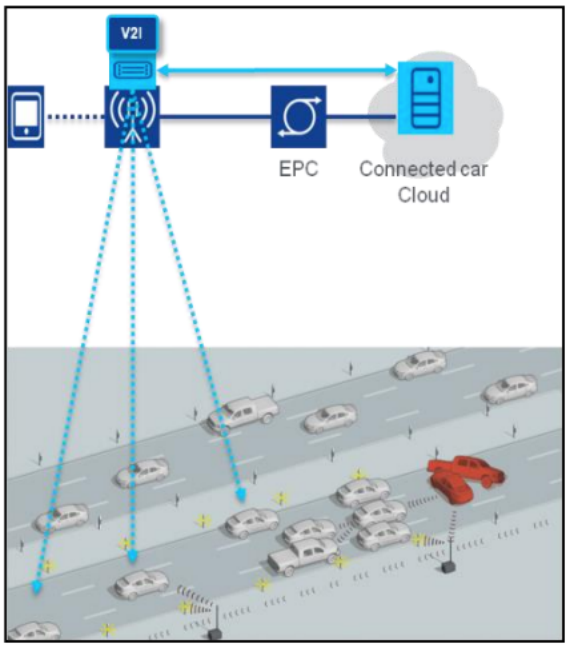

Figure 9 : Connected Vehicles Service Scenario

The roadside MEC application will be able to inform adjacent Mobile Edge Computing servers about the event(s) and in so doing, enable these servers to propagate hazard warnings to cars that are close to the affected area. The roadside application will be able to send local

information to the applications at the connected car cloud for further centralized processing and reporting.

3.1.4 Internet of Things Gateway

The Internet of Things (IoT) generates additional messaging on telecoms networks, and requires gateways to aggregate the messages and ensure low latency and security. Because of the nature of some of the devices being connected, a real time capability is required and a grouping of sensors and devices is needed for efficient service. IoT devices are often resource constrained in terms of processor and memory capacity. There is a need to aggregate various IoT device messages connected through the mobile network close to the devices. This also provides an analytics processing capability and a low latency response time.

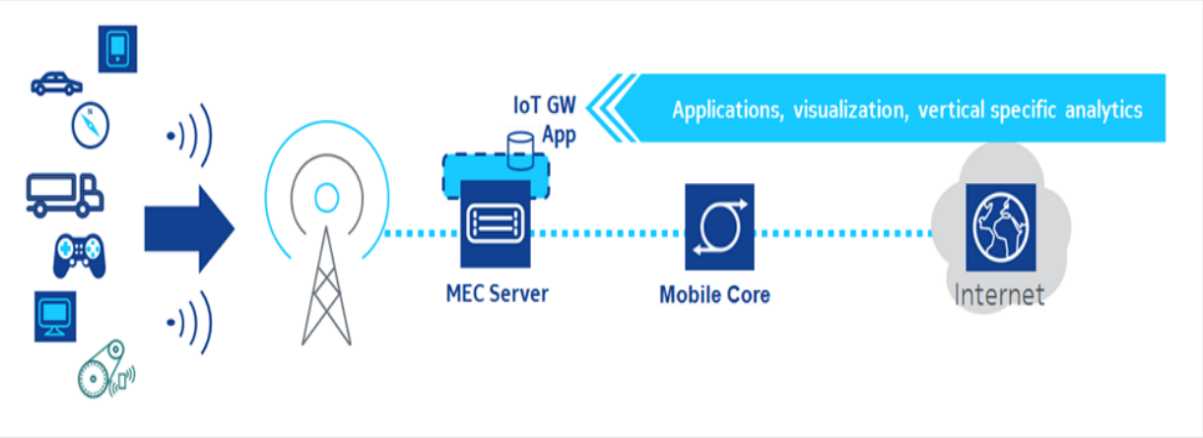

Figure 10: IoT Gateway Service Scenario

Various devices are connected over different forms of connectivity, such as 3G, LTE, Wi-Fi or other radio technologies. In general the messages are small, encrypted and come in different forms of protocols. There is a need for a low latency aggregation point to manage the various protocols, distribution of messages and for the processing of analytics. The MEC server provides the capability to resolve these challenges. Mobile Edge Computing can be used to connect and control devices remotely, analyse and provide real time provisioning and

analytics. MEC enables the aggregation and distribution of IoT services into the highly distributed mobile base station environment, and enable applications to respond in real-time. This can reduce the round trip time of data and enable a layer of abstraction from both the core network and applications in the cloud. IoT applications can run on MEC servers which are deployed at the LTE base station site to provide this functionality.

3.2 Description of the Reference Slicing Architecture

The overall latency budget of a client-server architecture on a Wireless Network may be split down into four main domains (see figure 11):

1. Access – the time required to go across the Radio Access Interface. Considering L3-L3 processing, typical access latency for legacy LTE is around 10-12ms. May be considerably reduced by 5G New Radio Access Interface.
2. Operator Network – the time required to pass through the Operator's Network, from the Radio Access Base Station up to the Internet Exchange point. It varies depending on the distance between the Client and the Gateway to the Internet Exchange Point, which can be up to many hundreds of km.
3. Internet – the time required to reach the final Server on the Internet. As it could actually be located anywhere in the world, this could be considerably high.
4. Processing – The time required by the server to decode the client request, to process it and to send back a response. This latency contribution is external to the Network and depends uniquely on SW architecture and on Processing Resources at the server side (thus the same should be considered on the client side as well).

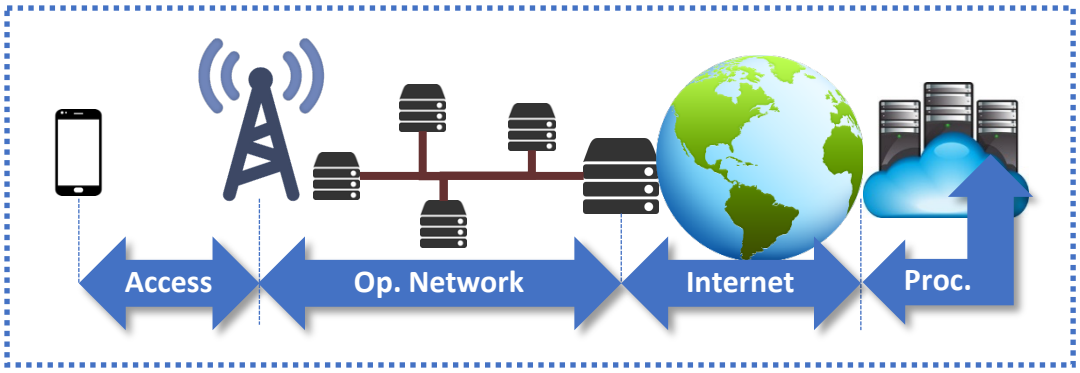

Figure 11. Domains of a Wireless Network

3.3 Multi-Access Edge Computing

The Base5G project assess the ability of MEC technology in reducing the impact of the 2nd and the 3rd contribution, while the 1st and the 4th are out of scope. The latency contribution given by the Internet may clearly be reduced by instantiating an Application Server as close as possible to the Internet Exchange point, e.g., co-located in the same server room with the Operator's border router. On the other hand, to reduce the contribution from the Operator's Network it is necessary for the Application Server to move further closer to the Client, i.e., to be enclosed within the Operator's Network itself.

LTE Enhanced Packet Core (EPC) embraced the new Virtualization paradigm and now many Operators are running in large part Virtual Core Network Functions, as this enables an easier deployment and a more elastic lifecycle management. 5G Core is based natively on VNF from its inception.

To further exploit the advantages of virtualization, the Rel 14 of the 3GPP introduces the concept of Control User Plane Separation (CUPS), which standardizes the interfaces between the Control Plane Functions and the User Plane Functions of the Serving Gateway and PDN Gateway (Figure 12). This makes it possible to physically deploy the two separate entities in different locations of the Networks. Full separation between Control Plane and User Plane Functions is a native feature of 5G Core.

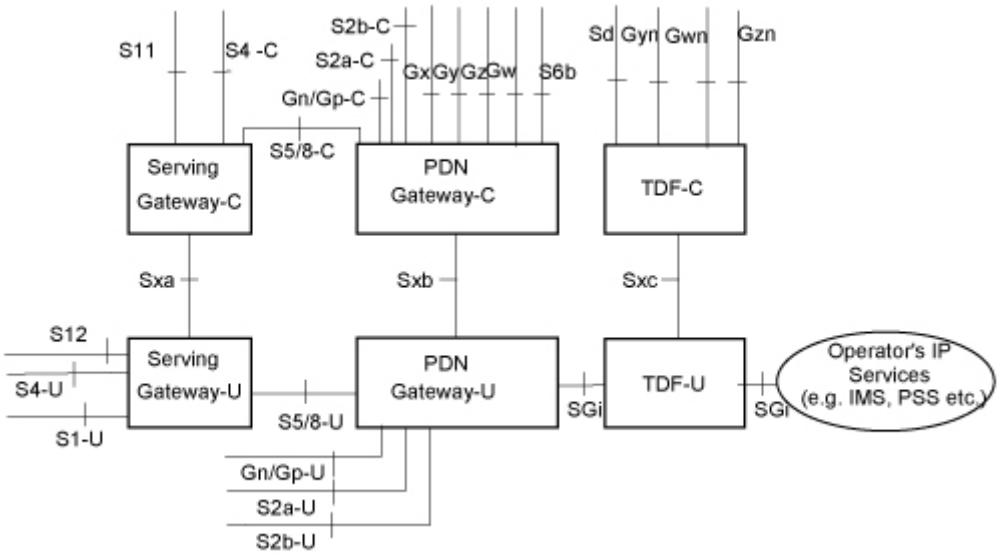

Figure 12. Architecture and interfaces of the CUPS

Virtualization and CUPS make it possible to distribute the User Plane Network Functions across multiple nodes in the network and to terminate the User Plane Traffic in peripheral locations instead of central ones when it is not necessary to reach a service deployed in remote location.

In order to deploy Services closer to the Client, the Operator must provision dedicated Compute, Storage and Networking resources within the network. These resources can be used to host User Plane VNFs and may be also used to host application VMs where Server functions are deployed and satisfy the client requests as close as possible to the Client itself.

Figure 13 shows the Base5G reference network, which comprises a breakdown of the Operator Network. The Operator Network comprises the Radio Access Nodes, the Security Gateways, the Core Network, and the Firewalls. It is connected to multiple Internet Exchanges that, through the Internet, make it possible to reach the Servers.

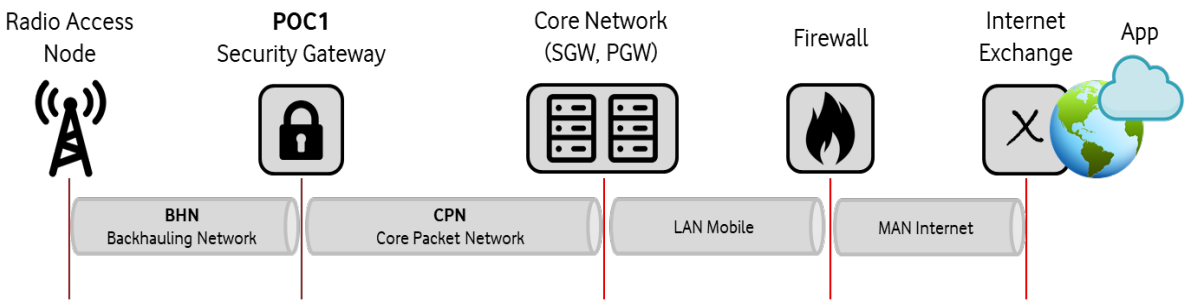

Figure 13. Segments and nodes in the Reference Network

MEC technology makes it possible to move the Application Server to some node internal to the Operator Network. A good candidate is the node that hosts the Security Gateway, which can be extended to host Application Servers and to terminate the traffic of those specific users that can uses the services delivered from the co-located Application Servers. Figure 14 shows the MEC-enabled Reference Network.

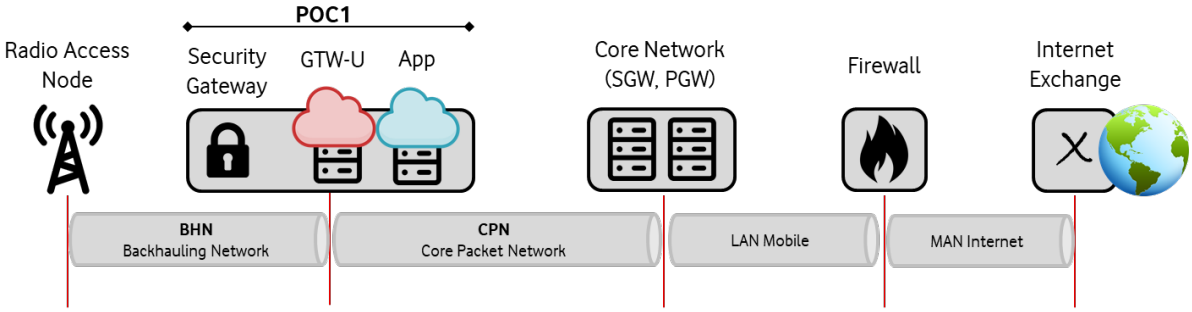

Figure 14. Reference Network with MEC and Application Servers co-located with the Security Gateway

Then, the MEC-enabled Reference Network can deliver services from multiple locations. We can identify two reference locations that only involve network segments under the control of the Operator. In the first case the Application is hosted on a datacenter co-located with the Central Core (A) rather than in many different datacenters each co-located with the Security Gateway (B).

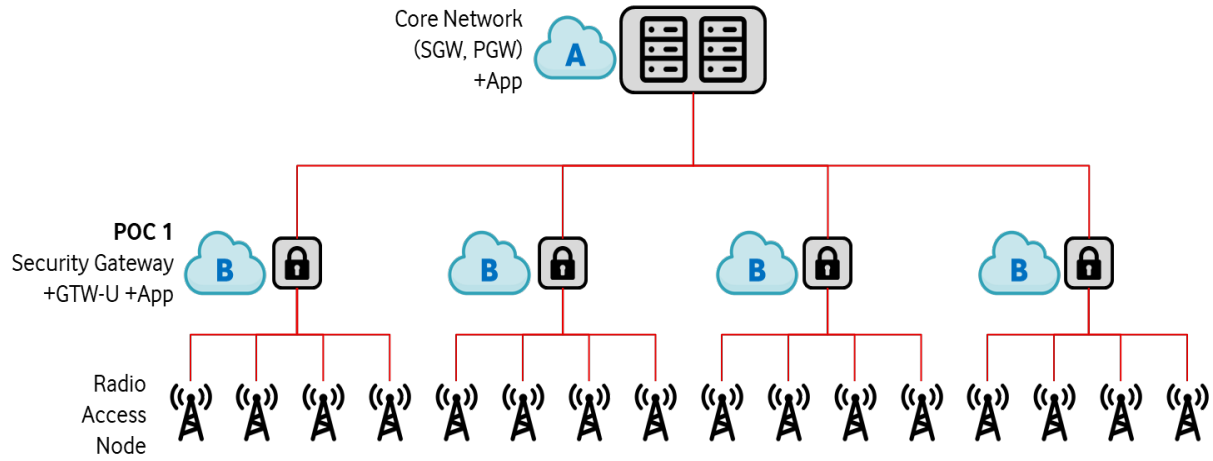

Figure 15. Reference Network with Application Serves in multiple locations: co-located with the Central Core Network (A) ore co-located with the Security Gateway (B).

3.4 On-Field Latency Measurements

In order to assess the advantages of MEC, we consider how the network latency changes with and without using MEC. The performance of the baseline solution with no MEC is evaluated by using latency measurements collected on the network of a Base5G partner.

Performance measurements are collected using the Two-Way Active Measurement Protocol (TWAMP) protocol (RCF5357), which defines a method for collecting two-way metrics and is supported by most vendors and is based on a client-server architecture. The TWAMP clients ("sender") sends a continuous flow of packets to a TWAMP server ("reflector"), which receives and reflects each packet back to the client.

The TWAMP client collects the Round-Trip-Time measurements, while the server only reflects the client packets as quickly as possible. The logical scheme of TWAMP is shown in Figure 16.

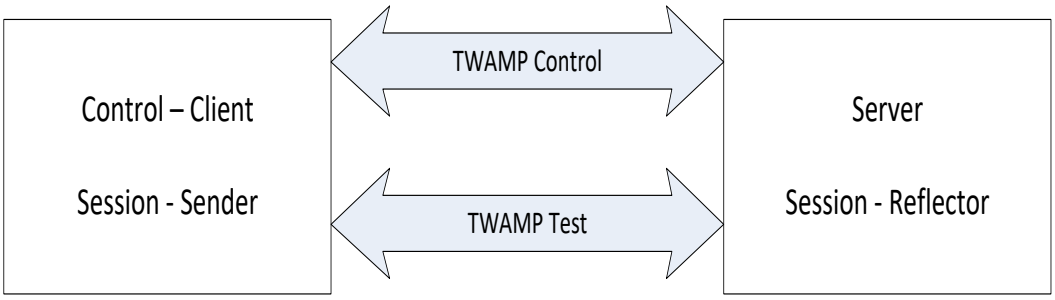

Figure 16. Logical scheme of the TWAMP protocol

The TWAMP client comprises:

- A Control-Client function, which sets up, starts, and stops the TWAMP test sessions.
- A Session-Sender function, which creates the TWAMP test packets sent to the Session-Reflector in TWAMP server.

The TWAMP server comprises:

- A Session-Reflector function, which sends back a measurement packet when a test packet is received but does not maintain a record of such information.
- A Server function, which manages one or more sessions with the TWAMP client and listens for control messages on a TCP port.

Figure 17 shows the architecture used to collect the data. The active probes working as sender nodes are deployed in any POC1 sites, in which they are co-located with the SecGw and the PE router, these nodes are the border network elements for the Backhauling and Core Packet networks, respectively.

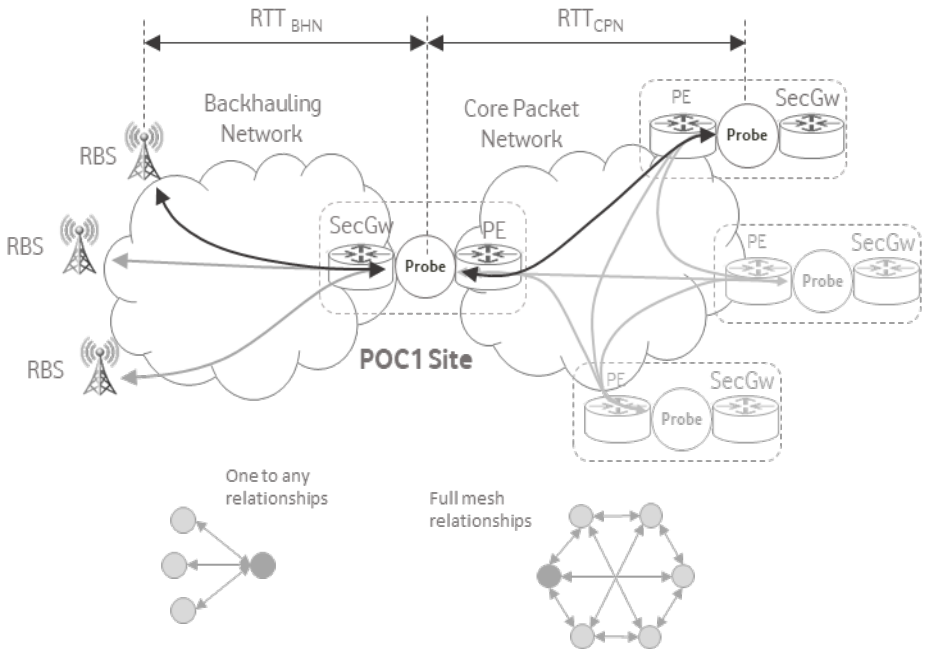

Figure 17. Measurement architecture

In the Backhauling Network (BHN), the probe compute measurements to the reflectors activated on each Radio Base Station in a one-to-many architecture. In the Core Packet Network, each probe works as sender and, at the same time, as reflector for the other probes. The resulting measurement architecture is a full mesh.

The session-sender starts transmitting UDP packets according to the send schedule and, upon arrival, the receiver timestamps the packets and stores the sequence number, send time, receive time and TTL from the IP header. This data will later be transferred back to the control-client for computation of performance metrics. The KPIs calculated from the TWAMP measurements are the Round Trip Time (RTT), the Jiitter, and the Frame Loss Ratio (FLR).

Each TWAMP session is characterized by the frequency with which the packets are sent, the packet size, and the Class of Service (CoS). Since the number of collected data points is high, especially for the BHN due to the considerable number of RBSes, data is aggregated before being used for any statistical analysis.

The basic level of aggregation starts from the Transmission Interval, defined previously as the reciprocal of the packet sending frequency. The Test Interval is the second hierarchical level and it represents the first level at which the single measurements captured by each individual packet transition are processed in terms of minimum, maximum and averages of all the measurements within the test interval. This represents the basic measure level. Starting from this level different aggregations are performed depending on the intended usage of the measurement itself. Base5G considers daily aggregations calculated as the 95th percentile of the measurements calculated over all the test intervals in a day. These ones, in turn, are calculated as the averages of all the measurements in each test interval. This process is represented in Figure 18.

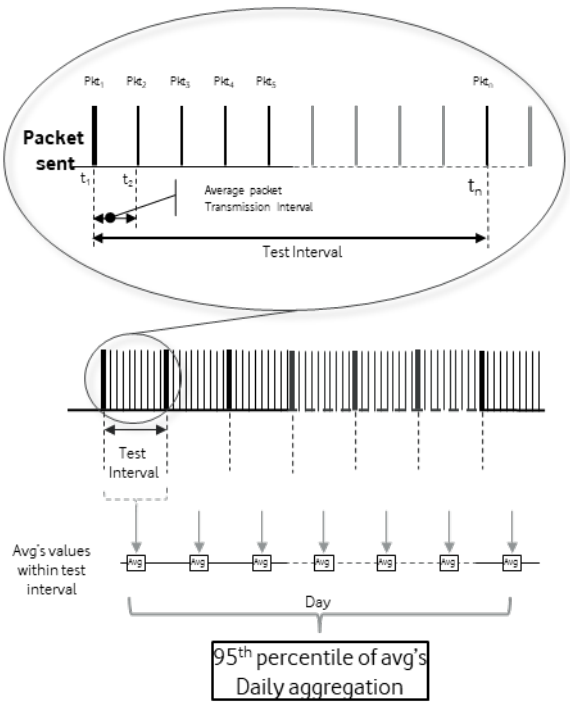

Figure 18. Calculation of daily measurements

For the sake of completeness, we also report that the measurements are collected using DSCP 46 as the Class of Service, which is associated to highest priority queue and thus not influenced by any traffic congestion. The test packet size is 200 bytes.

3.5 Cost Model

The costs of edge computing can vary wildly, depending on the size and scale of the deployment, the amount of data being collected and processed, and the geographic location of the edge computing deployment.

With reference to the Base5G architecture, we need to consider, for each MEC site, the costs to provide dedicated Compute, Storage, and Networking resources both to deploy the UP VNF and to host the VMs to run Server functions to answer the client requests as close as possible to the client itself.

The costs can be mainly differentiated between capital expenditures (CapEx) and operating expenses (OPEX) and both comprise fixed and variable costs. We consider the normalized costs in Table 2. Normalized costs for each site, where the different costs are normalized for a site that should managed a traffic of 10 Gbit/s.

Table 2. Normalized costs for each site

CapEx	Fixed Costs	Costs for the preparation of each new site and related to activities such as the implementation of power distribution, cooling systems and "building block" required for virtual infrastructure.	0.826
	Variable Costs	Costs related to the deployment of the virtual infrastructure (computing, storage, and networking). These costs mainly depend on the traffic managed by the	0.091 / 10 Gbit/s

		site.	
OPEX	Fixed Costs	Recurring costs for each site and per year after deployment include licenses of products with flat subscription fees.	0.075 /year
	Variable Costs	These costs typically depend on the number of deployed systems and therefore on the site traffic. Example of variable recurring costs are energy, maintenance, and licenses with subscription fees depending on number of systems. As for fixed OPEX costs, these costs should be considered for each site and year after deployment.	0.008 / 10 Gbit/s /year

CHAPTER FOUR

4 Optimization Model

Our reference scenario is shown in Figure 19. The network comprises:

- a Core Network, which is co-located to a datacenter hosting applications;
- a set G of Security Gateways;
- a subset M of the Security Gateways G that is MEC-enabled and hosts applications;
- a set R of Radio Access Nodes.

The Core Network and the Security Gateways are connected by means of an underlay network and form a full mesh. Each Radio Access Node is connected to one Security Gateway by means of a backhauling network.

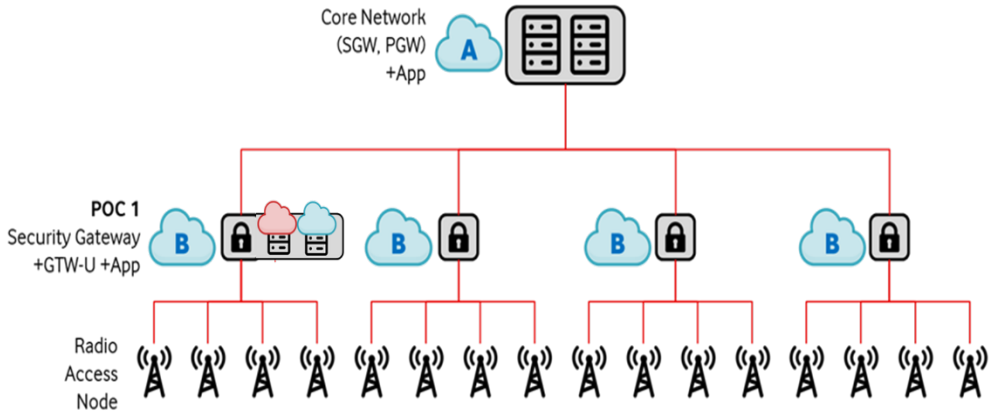

Figure 19. Reference Architecture for the techno-economical model

We introduce the following notation.

Let $r_{i,j}$ be the Round Trip Time (RTT) between the pair of gateways $i, j \in G$. For the sake of simplicity, we will label the Security Gateways with integer numbers 1, 2, … card(G).

We assume that the Core Network is collocated with the gateway with id $i = 1$, therefore $r_{1,j}$ represents both the RTT between gateway j and gateway i, but also between gateway j and the Core Network. Clearly $r_{i,j} = r_{j,i}$ and $r_{i,i} = 0$.

Let $b_{i,j}$ indicate whether the Radio Access Node i and the gateway j are directly connected. If $b_{i,j} = 1$, then the Radio Access Node i is directly connected to the gateway j. The converse is true if $b_{i,j} = 0$.

Let d_i be the RTT between the Radio Access Node i and the gateway to which it is connected.

Let $t_{i,j}$ be the RTT between the Radio Access Node i and the gateway j. If i is directly connected to j, then $t_{i,j} = d_i$, otherwise $t_{i,j} = d_i + r_{w,j}$, where w is the gateway to which i is connected.

Let m_i indicate whether gateway i is equipped with MEC. If $m_i = 1$, then the gateway i is equipped with MEC. The converse is true if $m_i = 0$.

Let D_i be the RTT between the Radio Access Node i and the nearest instance of the hosted application. We have that

$$D_i = \min t_{i,j} \text{ over all } j \text{ s.t. } m_j = 1 \text{ or } j = 1$$

The term j = 1 accounts for the RTT between the Radio Access Node and the application located in the Core Network.

4.1 Model 1

In the first model, we assume that the RTTs among the gateways, $r_{i,j}$, and the RTTs between each Radio Access Node and its connected gateway, d_i, are known. We also assume that there is a maximum number of MEC nodes that can be activated due to the budget constraint. The goal of the model is to find the location of the MEC nodes that results in the maximum number of Radio Access Nodes whose RTT to the closest instance of the edge application is below a given threshold.

We can formalize the model as follows, where M is the budget constraint and T is the delay threshold.

$$\max_i \delta_i$$

s.t.

$$\delta_i = \begin{cases} 0 & if\ D_i > T \\ 1 & if\ D_i \leq T \end{cases}$$

$$\sum_{j=1}^{\|G\|} m_j \leq M$$

There are two interesting extreme cases in this model. When M = 0, we have that all the RTTs are calculated w.r.t. the Core Network. When M=$\|G\|$, we have that all the gateways are equipped with MEC and thus, $D_i = d_i$ for all i.

4.2 Model 2

In the second model, we want that the RTT between each Radio Access Node and the closest application is below the threshold T and want to find the allocation that minimizes the cost. Consequently, we con formulate the problem as:

$$\min \sum_{j=1}^{\|G\|} m_j$$

s.t.

$$D_i \leq T \text{ for all } i$$

4.3 Models 3 and 4

The third and the fourth models are similar to the first and the second one, respectively, but with two important differences. First, each Radio Access Node is associated to an amount of traffic Λ_i. Second, the cost model of equipping a new MEC node comprises a fixed part, equal for all the gateways, and a variable part proportional to the traffic that flows through that gateway.

CHAPTER FIVE

5 Results and Discussions

As stated earlier, the goal of the work is to do a techno-economic analysis of Mobile Edge Computing using acquired data from Vodafone to evaluate the cost, in terms of the number of MEC-enabled sites, in order to provide different real-time applications.

Our raw data, gotten from Vodafone, was sampled couple of times and then the 95th percentile was recorded for each base-station site.

Firstly, data cleaning of our raw data was carried out, whereby the extremely high delays were deleted (all delays > 40ms). 12 rows had RTT values greater than 40ms, hence they were dropped

Initial number of sites – 19,067 sites
After data cleaning, total number of sites – 19,050 sites

We then show a preliminary application of our Models (see Chapter 4) by providing the analysis of the distribution of RTT between the Radio Access Node and the Application in 4 Scenarios, as mentioned in the objectives:

1. All the gateways have the data centre (in this case, the delay is the access delay)
2. Only gateway that has data centre is Milan (MI01)
3. MI01 + one additional gateway
4. Milan + any number of other gateways

The results of the analysis are presented in this chapter divided into different sections. Each section presents results of one analysis each.

5.1 Scenario 1 – All Gateways have MEC

The output of the table containing the RTT value for the first 5 sites is shown in the figure below:

Table 3: RTT values for the first 5 base-station sites in our datasheet

Out[4]:

	ZONE	REGION	SecGw SITE	Comune (SecGw Province)	RTT [ms]	DL_VOL [MB]	UL_VOL [MB]
0	1	LIGURIA	GE1	AIROLE (GE1)	4.776667	12306.305870	1154.833967
1	1	LIGURIA	GE1	ALASSIO (GE1)	2.456667	11655.466963	1091.590801
2	1	LIGURIA	GE1	ALASSIO (GE1)	2.671667	24967.335590	3089.475068
3	1	LIGURIA	GE1	ALASSIO (GE1)	2.691667	22670.992256	2254.333232
4	1	LIGURIA	GE1	ALASSIO (GE1)	2.726667	33814.876565	3651.312461

Next, 5 different reference round trip time (to accommodate different types of applications) were set, and the sites were evaluated based on these RTT value to get the number of sites that stay below these thresholds.

Reference RTTs set = 30ms, 20ms, 10ms, 5ms, and 2ms

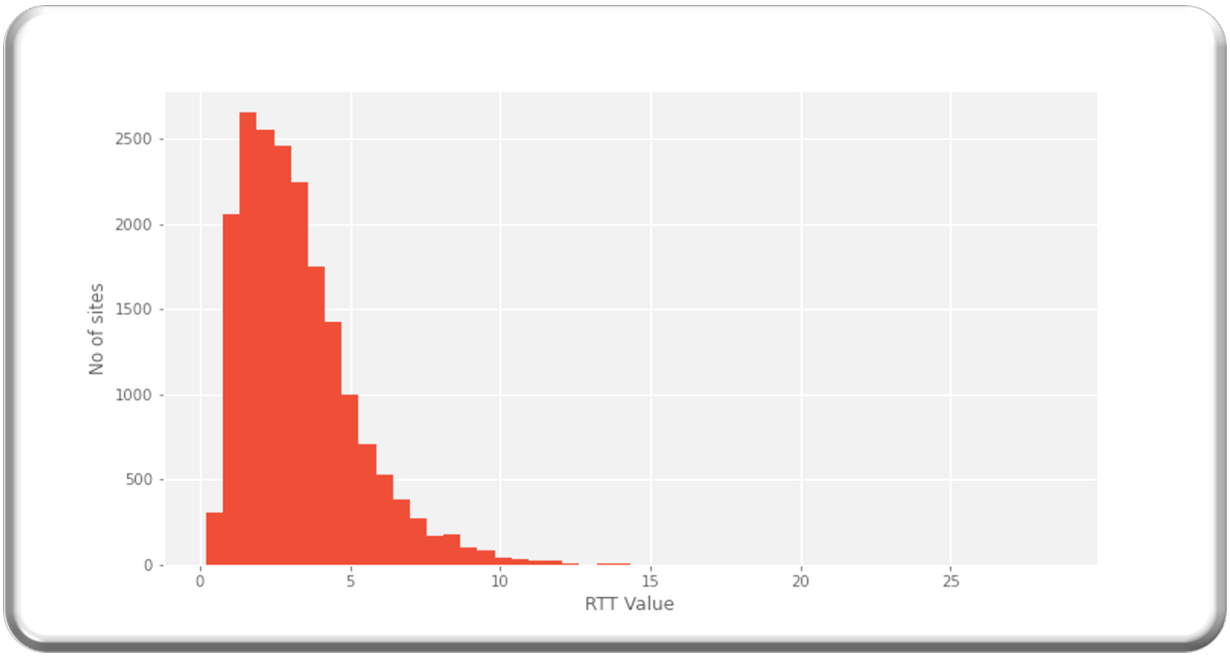

Figure 16. Frequency histogram for RTT from all sites to their region gateway (RTT threshold = 30ms)

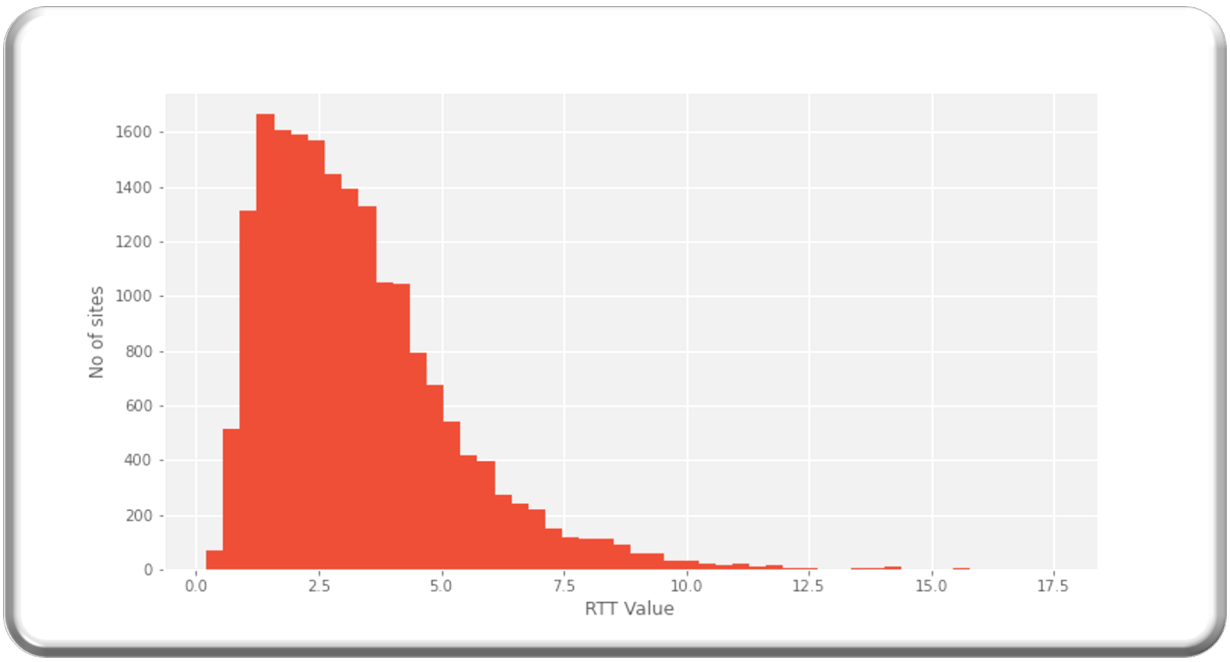

Figure 17. Frequency histogram for RTT from all sites to their region gateway (RTT threshold = 20ms)

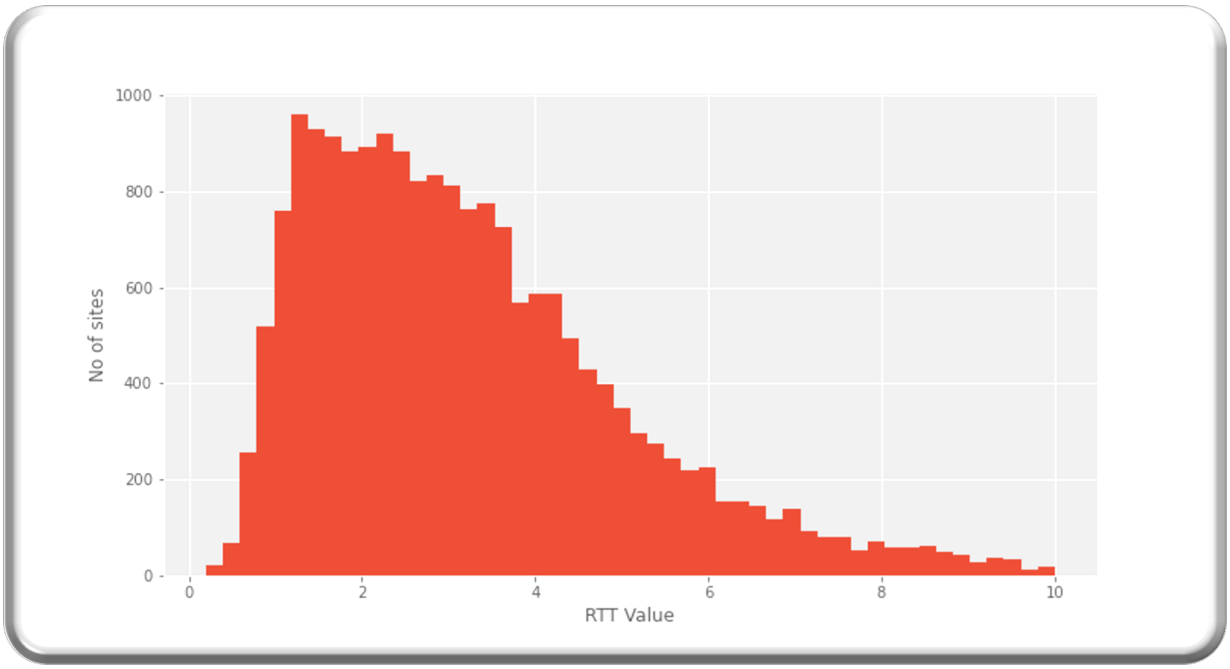

Figure 18. Frequency histogram for RTT from all sites to their region gateway (RTT threshold = 10ms)

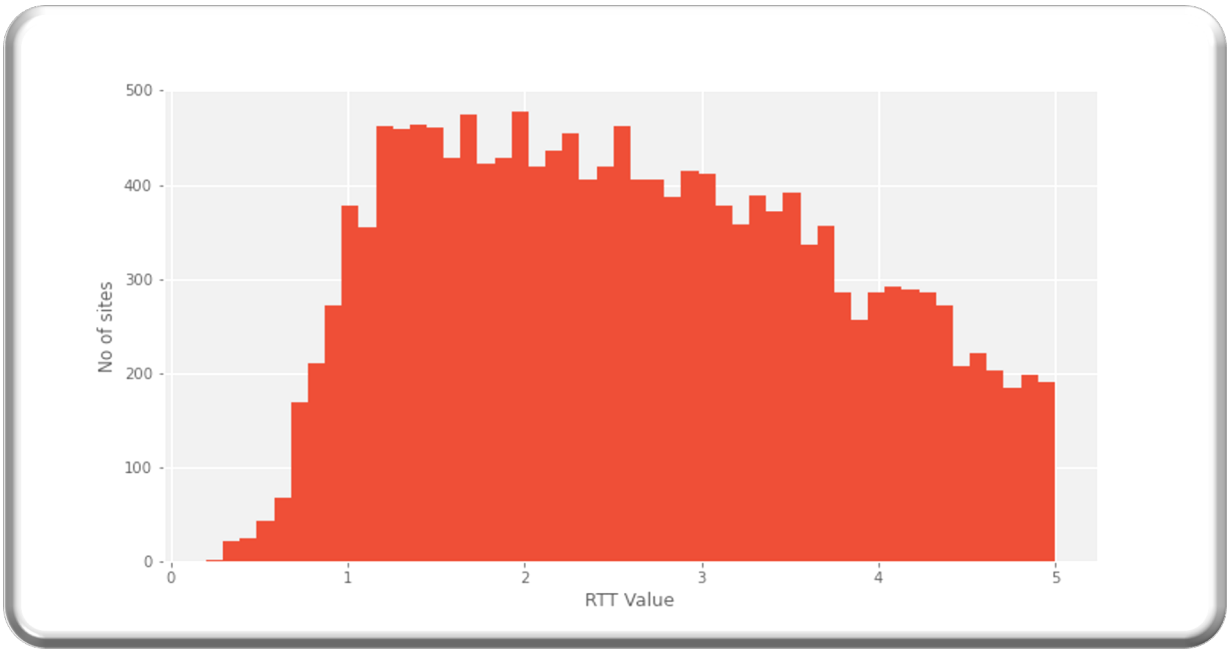

Figure 18. Frequency histogram for RTT from all sites to their region gateway (RTT threshold = 5ms)

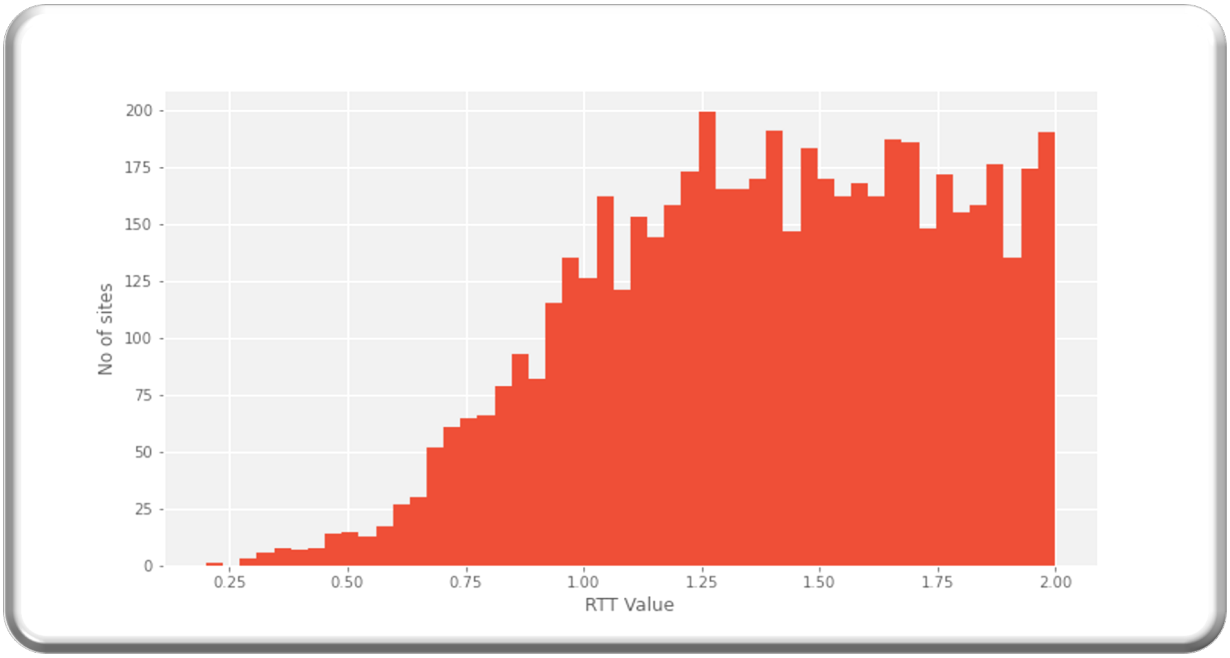

Figure 19. Frequency histogram for RTT from all sites to their region gateway (RTT threshold = 2ms)

5.2 Scenario 2 – Only Milan MI01 as Baseline Gateway

For each site, RTT to MI01 was calculated.

RTT to MI01 = [RTT access-to-gw] + [RTT gw-to-MI01]

```
Out[16]: (16442,
               site2MI01
         GE1    7.776667
         GE1    5.456667
         GE1    5.671667
         GE1    5.691667
         GE1    5.726667
         ..          ...
         BA2   28.712000
         BA2   28.906000
         BA2   29.602000
         BA2   28.328000
         BA2   28.982000

         [16442 rows x 1 columns])
```

Figure 20: output values of the calculated RTT to MI01 for all sites

Again, 5 different reference round trip time (to accommodate different types of applications) were set, and the sites were evaluated based on their RTT to MI01 value to get the number of sites that stay below these threshold

Reference RTTs set = 30ms, 20ms, 10ms, 5ms, and 2ms

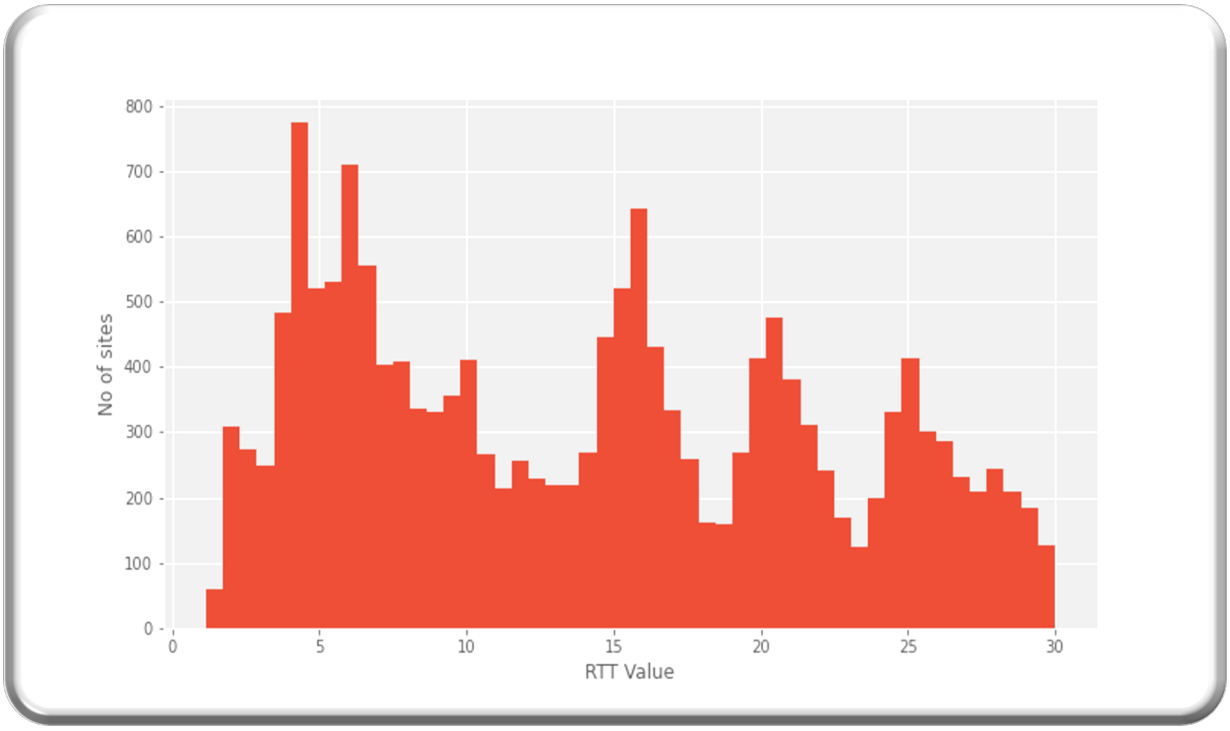

Figure 21: Frequency histogram for RTT from all sites to MI01 gateway (RTT threshold = 30ms)

16442 sites stay below this threshold

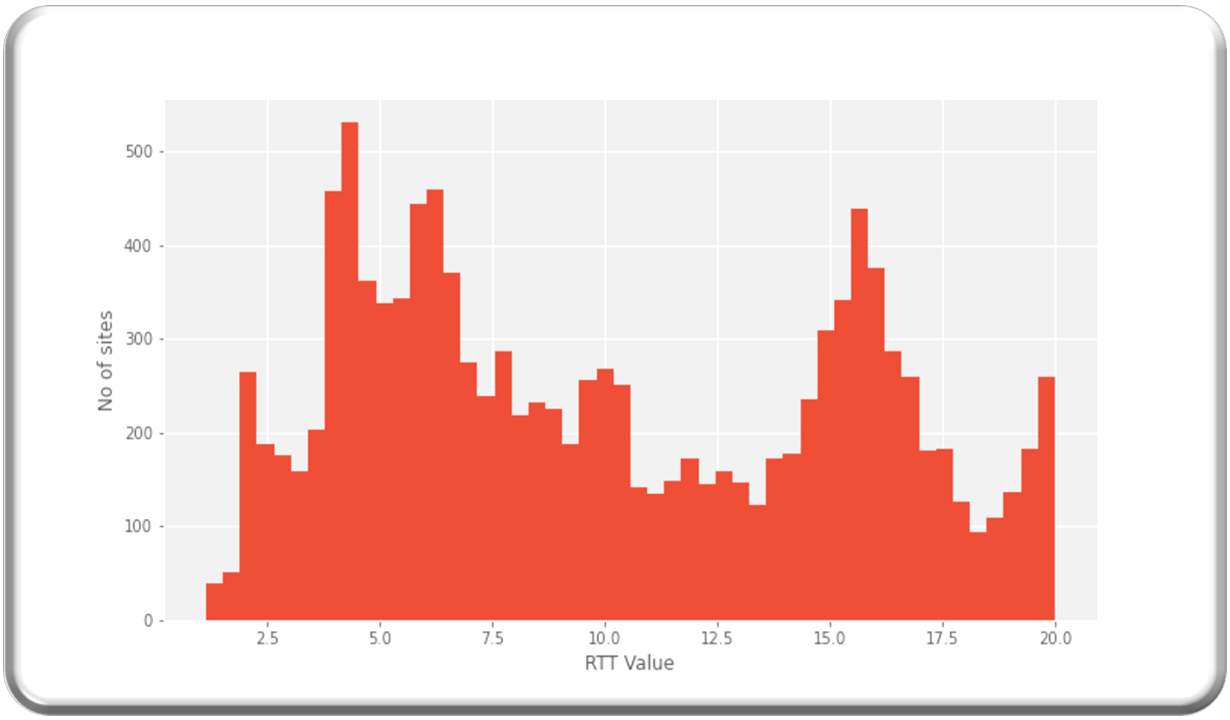

Figure 22: Frequency histogram for RTT from all sites to MI01 gateway (RTT threshold = 20ms)

11860 sites stay below this threshold

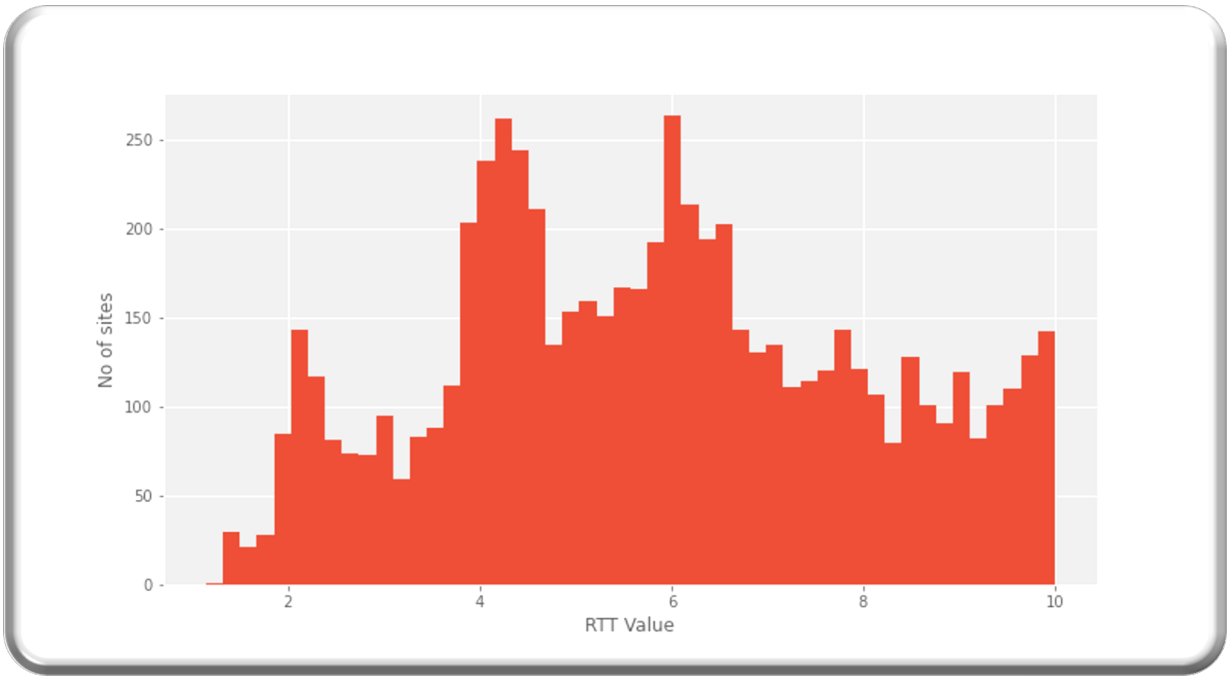

Figure 23: Frequency histogram for RTT from all sites to MI01 gateway (RTT threshold = 10ms)

6450 sites stay below this threshold

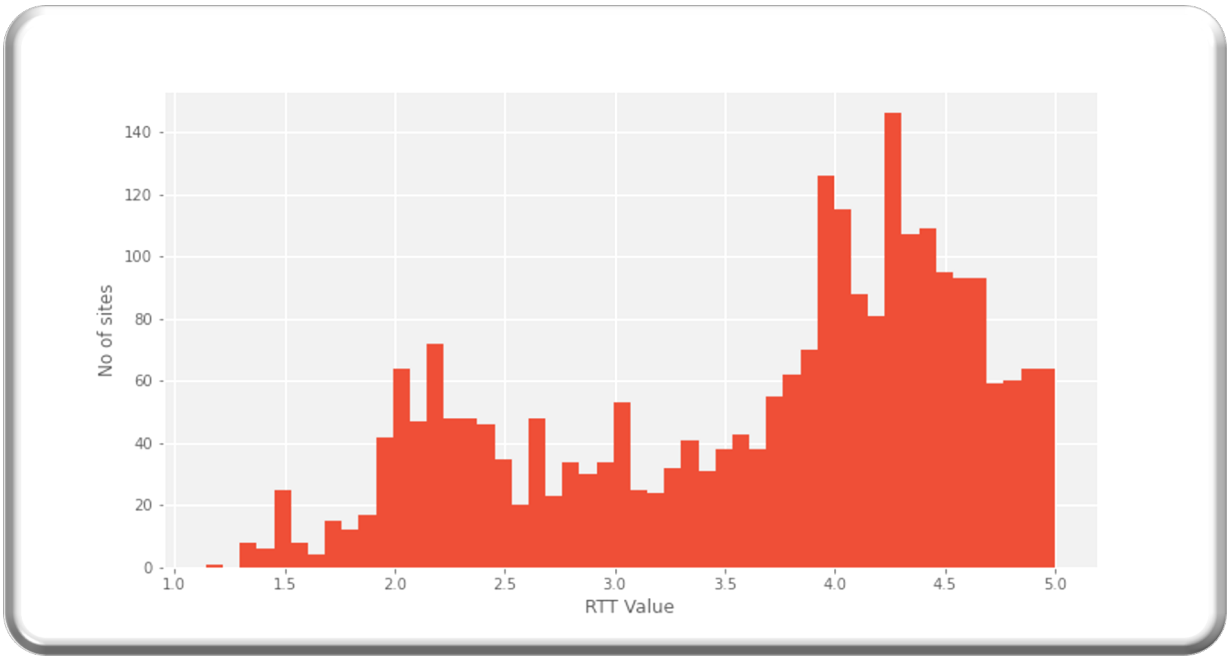

Figure 24: Frequency histogram for RTT from all sites to MI01 gateway (RTT threshold = 5ms)

2499 sites stay below this threshold

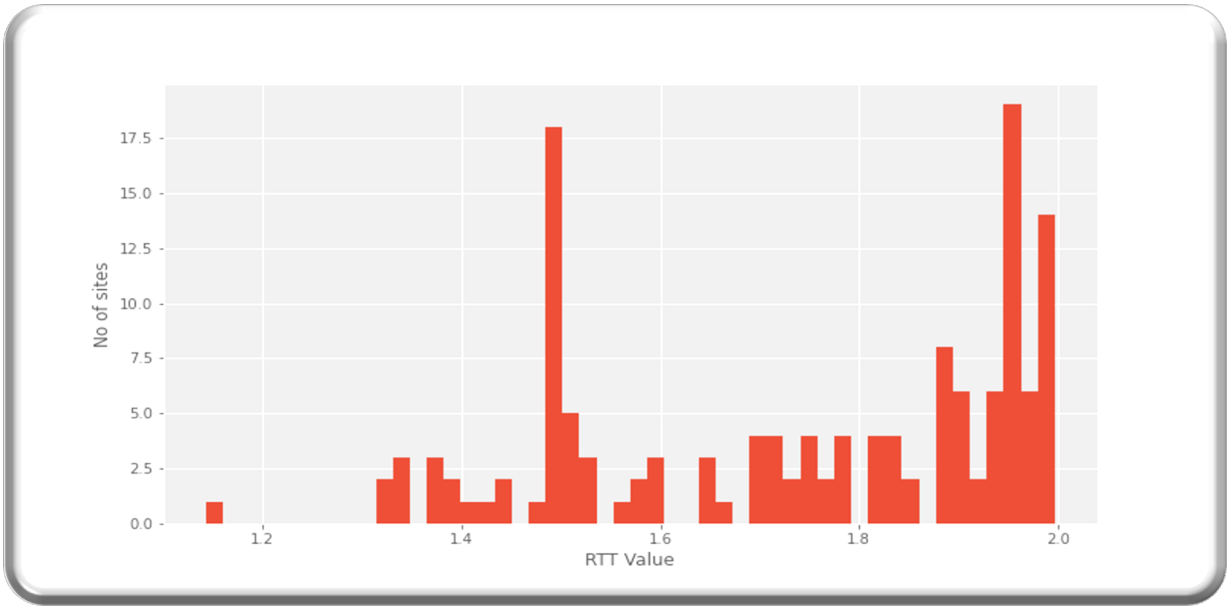

Figure 25: Frequency histogram for RTT from all sites to MI01 gateway (RTT threshold = 2ms)

143 sites stay below this threshold

5.3 Scenario 3 – Milan (MI01) + 1 additional gateway

For each site, RTT to each of the 30 gateways was calculated. For instance:

RTT to BG01 = [RTT access-to-gw] + [RTT gw-to-BG01]

Table 4: showing the RTT values of all the sites to each of the 30 gateways

	BG1	BS1	GE1	MI1	MI3	MI4	MI5	TO1	TO2	BO1	...	RM1	RM3	RM4	
GE1	12.986667	14.029792	4.776667	7.776667	7.906667	7.646667	7.946667	9.866667	9.876667	12.942667	...	18.646667	20.956667	20.716667	30.11
GE1	10.666667	11.709792	2.456667	5.456667	5.586667	5.326667	5.626667	7.546667	7.556667	10.622667	...	16.326667	18.636667	18.396667	27.79
GE1	10.881667	11.924792	2.671667	5.671667	5.801667	5.541667	5.841667	7.761667	7.771667	10.837667	...	16.541667	18.851667	18.611667	28.01
GE1	10.901667	11.944792	2.691667	5.691667	5.821667	5.561667	5.861667	7.781667	7.791667	10.857667	...	16.561667	18.871667	18.631667	28.03
GE1	10.936667	11.979792	2.726667	5.726667	5.856667	5.596667	5.896667	7.816667	7.826667	10.892667	...	16.596667	18.906667	18.666667	28.06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
PA1	52.676000	48.155688	49.526000	47.346000	44.896000	42.786000	43.076000	49.346000	49.446000	42.496000	...	39.096000	29.866000	29.706000	40.25
PA1	47.390000	42.869688	44.240000	42.060000	39.610000	37.500000	37.790000	44.060000	44.160000	37.210000	...	33.810000	24.580000	24.420000	34.97
PA1	47.513333	42.993021	44.363333	42.183333	39.733333	37.623333	37.913333	44.183333	44.283333	37.333333	...	33.933333	24.703333	24.543333	35.09
PA1	47.596000	43.075688	44.446000	42.266000	39.816000	37.706000	37.996000	44.266000	44.366000	37.416000	...	34.016000	24.786000	24.626000	35.17
PA1	48.068333	43.548021	44.918333	42.738333	40.288333	38.178333	38.468333	44.738333	44.838333	37.888333	...	34.488333	25.258333	25.098333	35.64

19050 rows × 31 columns

A matrix was then formed with the rows being the access sites and the column being the gateways

```
[[13.21666667 14.02979167  4.77666667 ... 30.45666667 29.10666667
  46.93666667]
 [10.89666667 11.70979167  2.45666667 ... 28.13666667 26.78666667
  44.61666667]
 [11.11166667 11.92479167  2.67166667 ... 28.35166667 27.00166667
  44.83166667]
 ...
 [47.51333333 42.99302083 44.39333333 ... 21.37333333 20.11333333
   2.20333333]
 [47.596      43.0756875  44.476      ... 21.456      20.196
   2.286     ]
 [48.06833333 43.54802083 44.94833333 ... 21.92833333 20.66833333
   2.75833333]]
Shape of the data matrix: (19050, 31)
```

Figure 26: Matrix form of the access sites and the gateways

Based on the RTT values gotten, a threshold RTT value was set and the number of sites that stay below this threshold was calculated.

Assumption:

Threshold RTT chosen = 30ms

Then, the number of sites that stay below 30ms were maximised by building a new matrix such that it marks:

- 1, if below the threshold and,
- 0, if above

Table 5: New matrix representation that marks 1 for sites below threshold, and 0 for sites above threshold

	BG1	BS1	GE1	MI1	MI3	MI4	MI5	TO1	TO2	BO1	...	RM1	RM3	RM4	BA1	BA2	CT2	CZ2	NA1	NA2	PA1
0	1	1	1	1	1	1	1	1	1	1	...	1	1	1	0	0	0	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
2	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
3	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
4	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19045	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
19046	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
19047	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
19048	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
19049	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1

19050 rows × 31 columns

In order to choose which additional gateway is the most preferred, 2 different analysis models are considered:

1. **Highest number of sites covered:** With this method, depending on our set RTT threshold, we sum up the values of ones and zeroes for each gateway (column-wise) in our matrix. Then we choose the gateway (column) with the highest sum as our additional gateway, This is because this is the gateway that maximizes the number of sites that stay below our set threshold.

However, it should be noted that although this analysis method works, it was not an optimal method.

Table 6: Using highest number of sites covered method to output the additional gateway choice and the number of sites covered

	BG1	BS1	GE1	MI1	MI3	MI4	MI5	TO1	TO2	BO1	...	RM1	RM3	RM4	BA1	BA2	CT2	CZ2	NA1	NA2	PA1
GE1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	0	0	0	0	0	1	0
GE1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
GE1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
GE1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
GE1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	0	0	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
PA1	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
PA1	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
PA1	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
PA1	0	0	0	0	0	0	0	0	0	0	...	0	1	1	0	1	0	0	1	1	1
SUM	14204	15534	15688	16442	15805	15890	15740	16012	15716	17398	...	17391	18926	18940	13855	13894	4922	6207	16479	16903	3950

19051 rows × 31 columns

Using this analysis method, and a set threshold of 30ms, the additional gateway that is chosen to be activated as an MEC with MI1 is 'RM4' since it had the highest sum with a coverage of 18940 sites.

2. **Best coverage combination with MI01**: This particular analysis shows the combination of gateway that complements each other to get the best highest number of coverage, unlike in the previous analysis that shows only the highest number of coverage for a single gateway. Simply put, this analysis looks for the best gateway that complements MI10 in a way that it tries to cover the zeroes of MI1 in the matrix with a one, such that if this gateway is chosen as the additional gateway, activating the combination of this gateway with MI01 will give us the best highest number of coverage.

This is an optimal method, and was the analysis method chosen for the of this scenario and the next scenario (scenario 4)

The additional gateway chosen using of the best coverage combination model is **'NA2'.** Activating the combination of the additional gateway, NA2, with MI1, covers a total of **19038 sites**.

This is a much better coverage as compared to the previous analysis method where only the additional gateway chosen covered only 18940 sites

Next, the column for RTT to MI01, and the column for the RTT of the chosen additional gateway (in this case NA2) are compared, and the delay is calculated as the minimum between these 2 columns (gateways)

The frequency histogram for the selected RTT for all sites, based on the specified 30ms threshold, is then created.

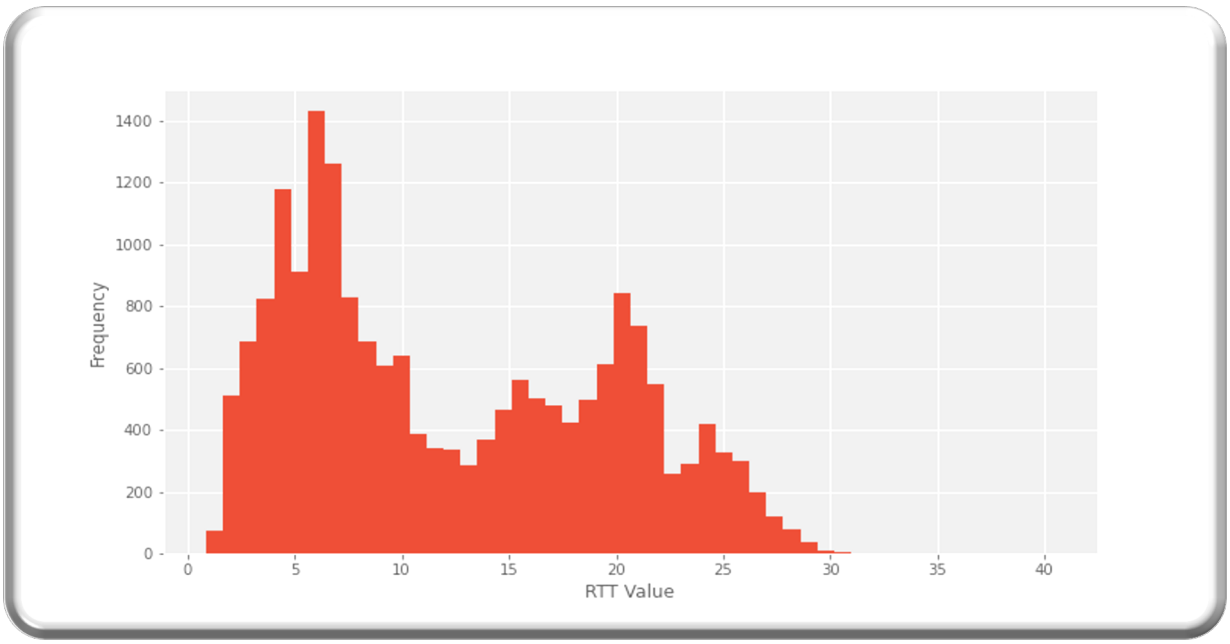

Figure 27: Frequency histogram of the selected RTT for all sites (with RTT threshold = 30ms)

Next, the data was analysed using different set RTT threshold values (20ms, 10ms, 5ms, and 2ms). For each of these values, result changes as summarised below:

Table 7: Result summary of the analysis of the two different methods above, while varying RTT threshold values

Threshold (ms)	Additional Gateway (Analysis 1)	No of sites covered	Additional Gateway (Analysis 2)	No of sites covered
20	RM4	14381	FI1	16054
10	BO1	6720	FI1	9421
5	MI4	2588	BO1	4211
2	MI4	558	RM3	658

Now, using the preferred method (analysis 2) as explained while considering RTT threshold of 30ms, the different histograms were then plotted accordingly

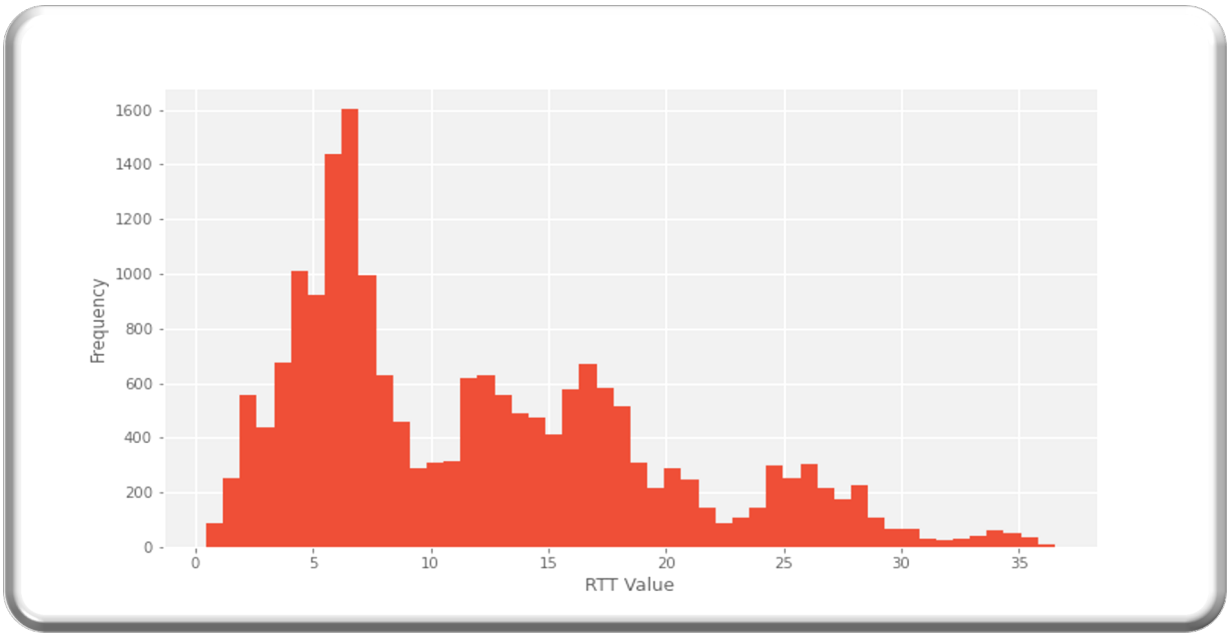

Figure 28: Frequency histogram of the selected RTT for all sites (with RTT threshold = 20ms)

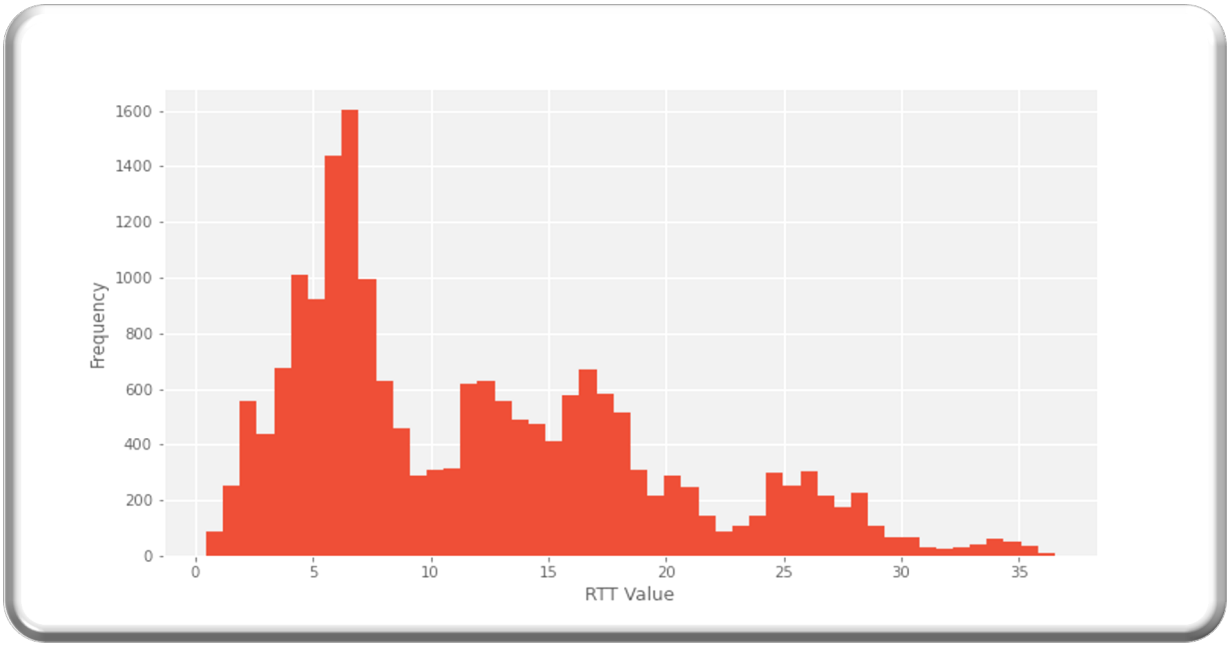

Figure 29: Frequency histogram of the selected RTT for all sites (with RTT threshold = 10ms)

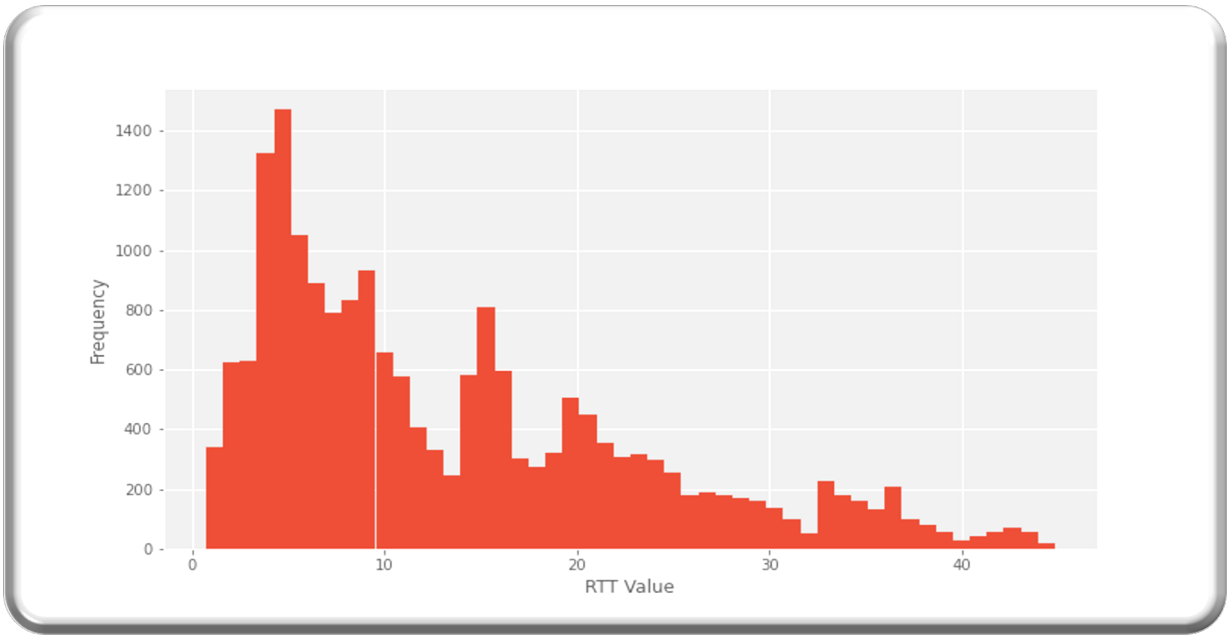

Figure 30: Frequency histogram of the selected RTT for all sites (with RTT threshold = 5ms)

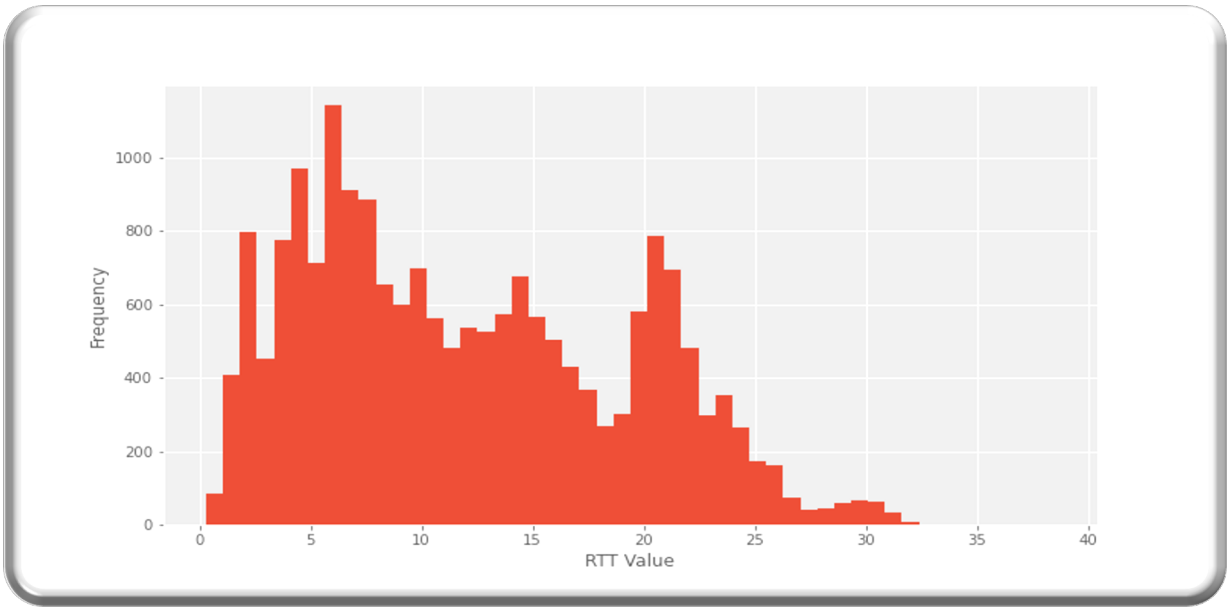

Figure 31: Frequency histogram of the selected RTT for all sites (with RTT threshold = 2ms)

5.4 Scenario 4 – Milan (MI01) + Any Number of Additional Gateways

Using the best coverage method, the data was analysed for this scenario using different set RTT threshold values (30ms, 20ms, 10ms, 5ms, and 2ms). As expected, for each of these RTT values, the result changes and can be summarised in Table 7 below:

Table 8: Analysis of scenario 4 using best coverage method, which outputs the additional gateways to be chosen as MEC for the varying threshold values

Threshold (ms)	2 Additional Gateway and the no of sites covered	3 Additional Gateways and the no of sites covered	4 Additional Gateways and the no of sites covered
30	NA2 : 19038, FI1: 19044	NA2 : 19038, FI1: 19044, BA1 : 19049	NA2 : 19038, FI1: 19044, BA1 : 19049, AN2 : 19050
20	FI1: 16054, NA2 : 17018	FI1: 16054, NA2 : 17018, CZ2 : 17779	FI1: 16054, NA2 : 17018, CZ2 : 17779, TS1 : 18191
10	FI1: 9421, NA2 : 11240	FI1: 9421, NA2 : 11240, VR1 : 12265	FI1: 9421, NA2 : 11240, VR1 : 12265, CT2 : 13128
5	BO1 : 4211, RM4 : 5308	BO1 : 4211, RM4 : 5308, NA2 : 6290	BO1 : 4211, RM4 : 5308, NA2 : 6290, VR1 : 7028
2	RM3 : 658, BO2 : 1103,	RM3 : 658, BO2 : 1103, TO2 : 1527,	RM3 : 658, BO2 : 1103, TO2 : 1527, MI4 : 1943

Now, using the preferred model (analysis 2 - best coverage) as explained earlier in scenario 3, and setting the number of additional gateways needed as 4, the histograms for the different threshold values were then plotted accordingly

```
In [66]: ## Multiple gateways selection
         number = 4
         gateways = ScenarioC_BestCoverage(number,gateway_matrix)
         gateways

Out[66]: {'NA2': 19038, 'FI1': 19044, 'BA1': 19049, 'AN2': 19050}

In [67]: MulitipleSelectedGatewayHist(gateways,gateway_dist,threshold,50)
```

Figure 32: showing how the number of additional gateways needed is inputed in our code and the output it gives

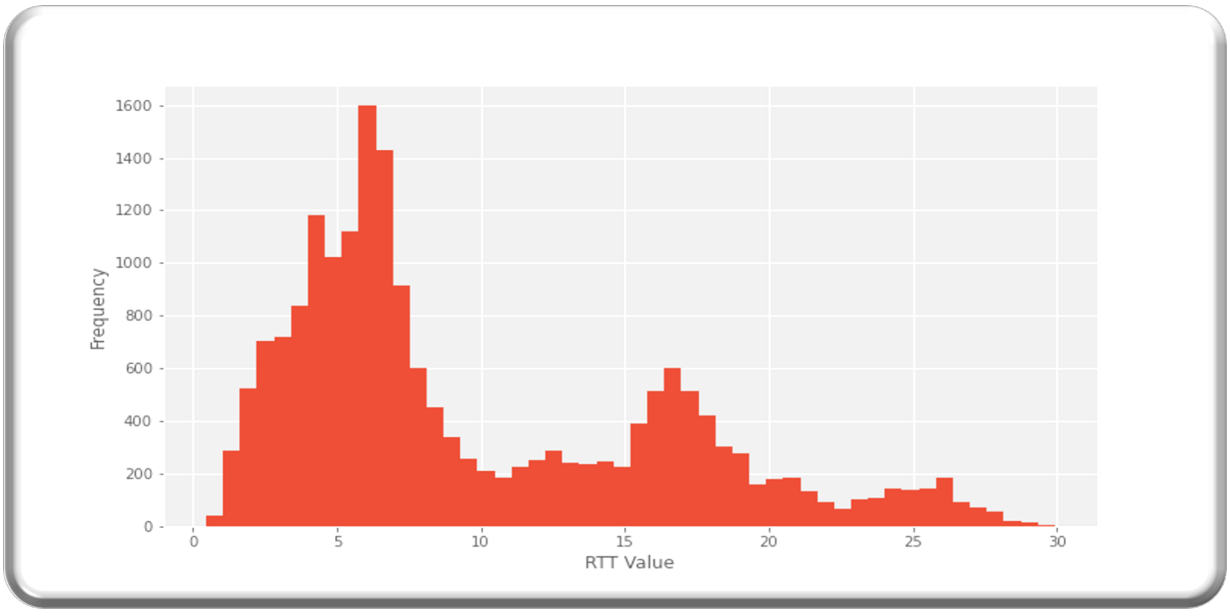

Figure 33: Frequency histogram of the selected RTT for all sites (with RTT threshold = 30ms)

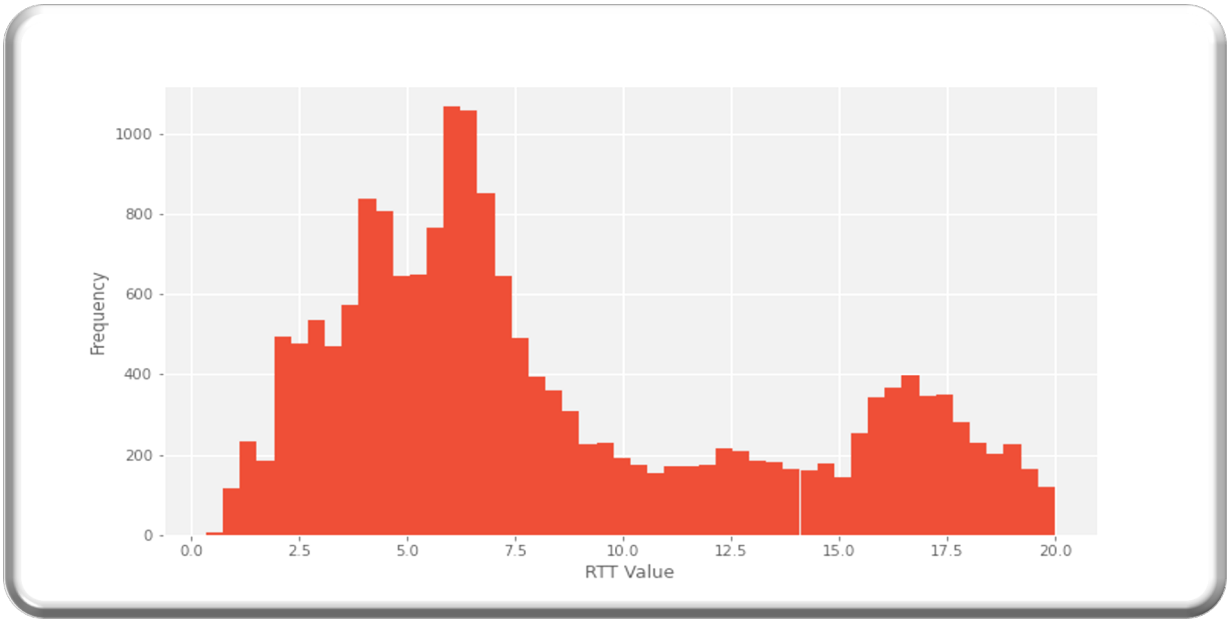

Figure 34: Frequency histogram of the selected RTT for all sites (with RTT threshold = 20ms)

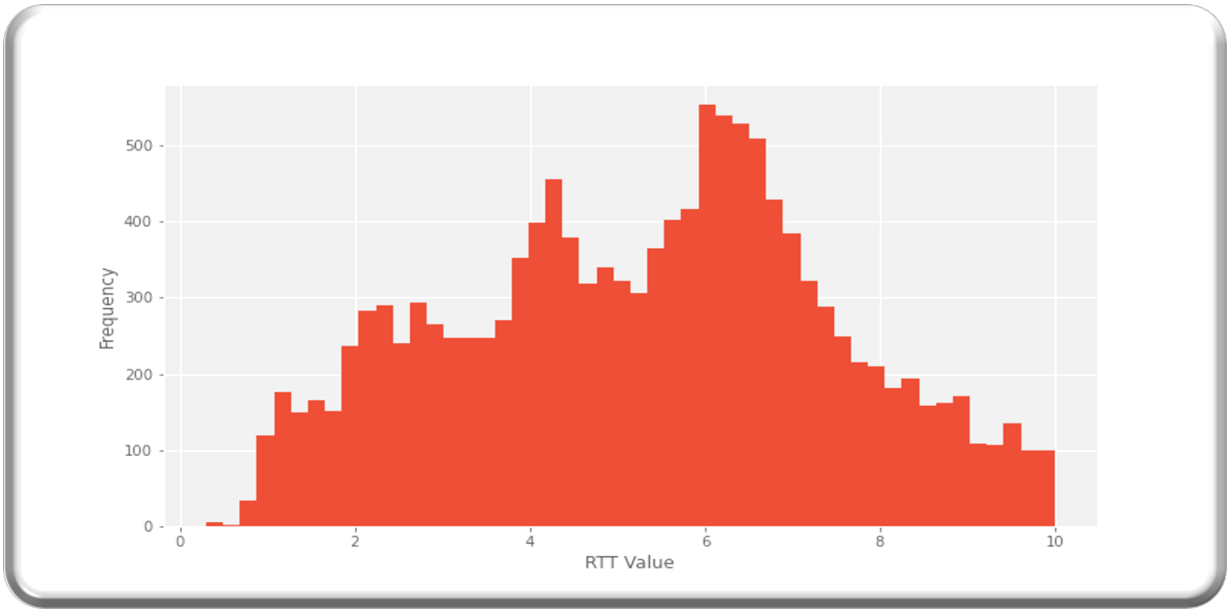

Figure 35: Frequency histogram of the selected RTT for all sites (with RTT threshold = 10ms)

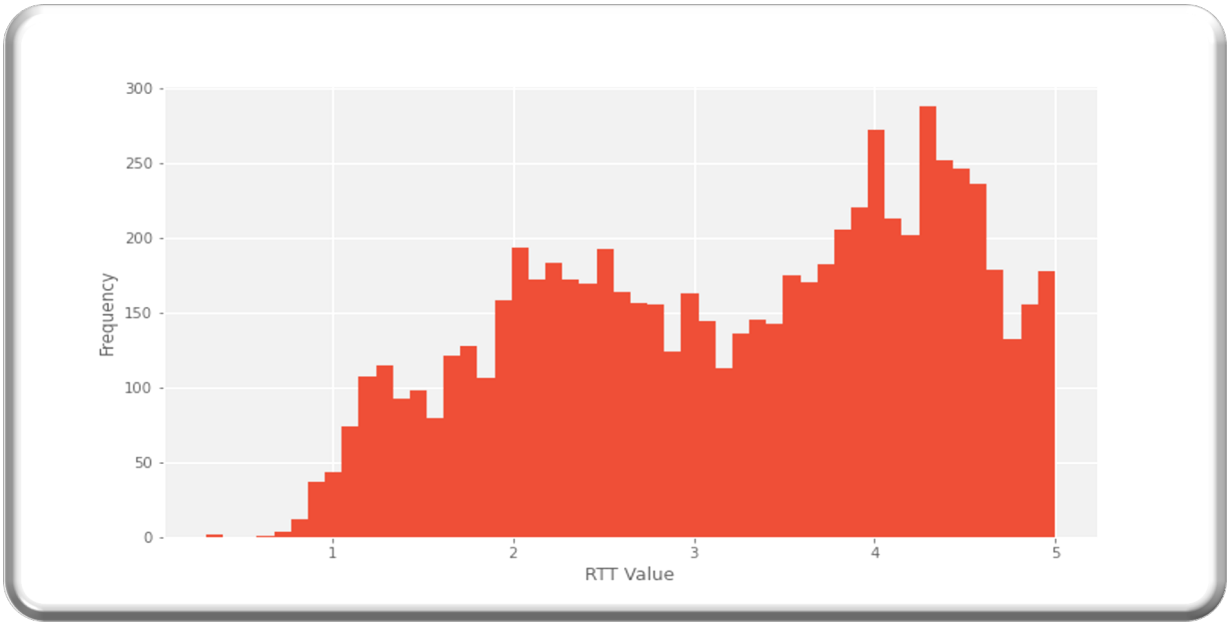

Figure 36: Frequency histogram of the selected RTT for all sites (with RTT threshold = 5ms)

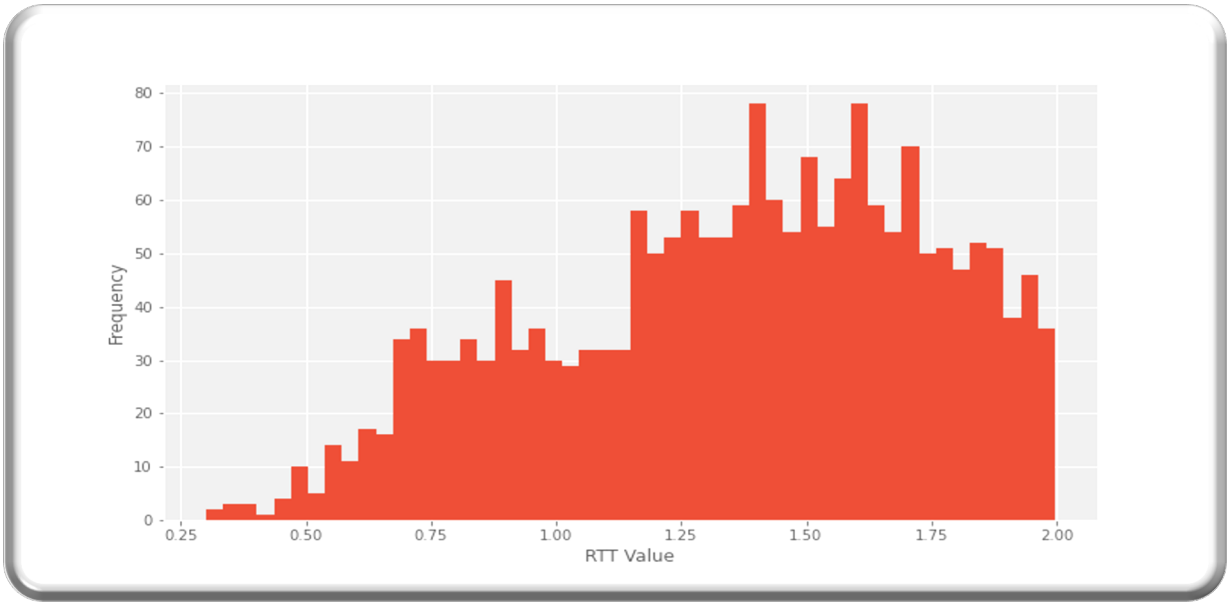

Figure 37: Frequency histogram of the selected RTT for all sites (with RTT threshold = 2ms)

5.5 Calculation of total cost budget

Assuming that the different gateways have different costs, and that we know how many gateways to activate as MEC in order to maximise site coverage that falls below a given threshold, we want to calculate the cost (i.e how much money is needed as budget)

To do this, we first randomly assign cost to all the different gateways.

```
In [159]: #randomly assign cost to the different gateways
          gateways = SecGw_red.index
          costs = {}
          for i in gateways:
              costs[i]=random.randint(10,20)

In [42]: print(costs)

         {'BG1': 16, 'BS1': 14, 'GE1': 19, 'MI1': 14, 'MI3': 12, 'MI4': 10, 'MI5': 10, 'TO1': 18, 'TO2': 20, 'BO1': 19, 'BO2':
         13, 'PD1': 19, 'PR1': 19, 'TS1': 18, 'VE1': 11, 'VR1': 12, 'AN2': 15, 'CA1': 11, 'FI1': 18, 'PE1': 18, 'PI2': 12, 'RM
         1': 13, 'RM3': 18, 'RM4': 14, 'BA1': 11, 'BA2': 13, 'CT2': 20, 'CZ2': 19, 'NA1': 14, 'NA2': 20, 'PA1': 11}
```

Figure 38: Assigning costs randomly to the different gateways

Next, we input the number of additional gateways we want to use (say 4 gateways for instance). After which we now calculate the costs of all the selected gateways by taking the sum

Getting Cost for selected gateways

```
In [160]: def cal_cost_gateways(best_gateways, costs):
              sum = 0
              for i in best_gateways:
                  sum+=costs[i]

              return sum

In [161]: #calculating cost of selected gateways
          number = 4
          best_gateways = ScenarioC_BestCoverage(number,gateway_matrix)

          #pass in best_gateways dictionary and weights dictionary
          sum = cal_cost_gateways(best_gateways,costs)
          sum

Out[161]: 60
```

Figure 39: Getting costs for selected gateways

5.6 Calculating Best Coverage Under Budget

Here, we are trying to find the maximum number of gateways that will give the best coverage of sites assuming we have a given budget. Apart from knowing the number of gateways, we also want to know the gateways that will be activated, the coverage, and the total cost (which should be less than or equal to the given budget

To achieve this,

1. All the possible combinations of gateways were calculated such that the total sum of the cost is less than or equal to the budget.
2. Based on the combinations found, the coverage of each gateway combinations is then calculated and then the gateway combination with the highest coverage and the lowest cost is selected

Assuming that we were given a budget of 50 for instance, it can be observed that activating a total of 3 gateways (MI1 + 2 additional gateways) will give the best coverage of sites (19044 out of 19050) with a total cost of 41, which is below the given budget.

```
In [167]: Budget = 50
          optimized_value = Cal_OptimumGatewayCombination(costs,Budget,gateway_matrix,"MI1")
          print(optimized_value)

          ({'coverage': 19044}, {'gateways': ['MI1', 'FI1', 'NA2']}, {'Total cost': 41})
```

Figure 40: Assigning a random budget to get the resulting coverage, additional gateways, and total cost

5.7 Calculation based on traffic volume (FOR SCENARIO 1)

So far, we have considered using just RTT values for our analysis. Now, we shall make our calculations based on the traffic volume.

A threshold value was set for the upward and downward traffic volume respectively, the data was then analysed based on these set threshold, and finally the histogram and bar chart was plotted to determine the number of sites that fall above the set traffic threshold. This threshold values set can be varied depending on the application in use

Assumption:

Upload traffic Volume Threshold chosen = 3000MB

Download traffic Volume Threshold chosen = 40000MB

With Upload traffic Volume Threshold of 3000MB

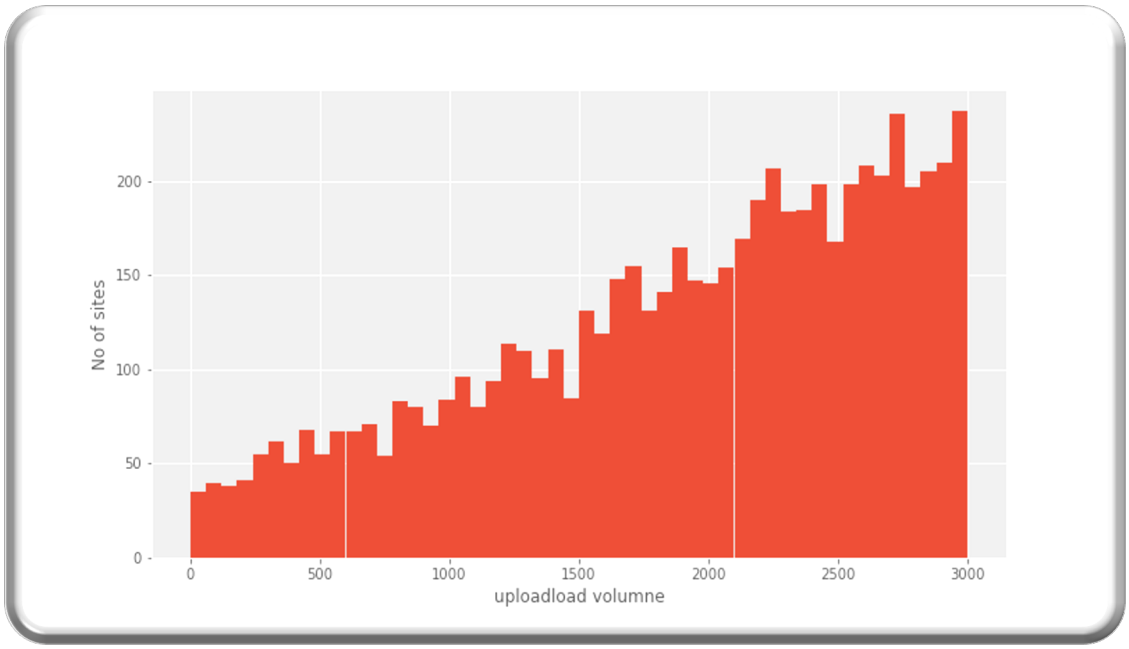

Figure 41: Histogram showing the number of sites that fall above the set upload traffic volume threshold of 3000MB

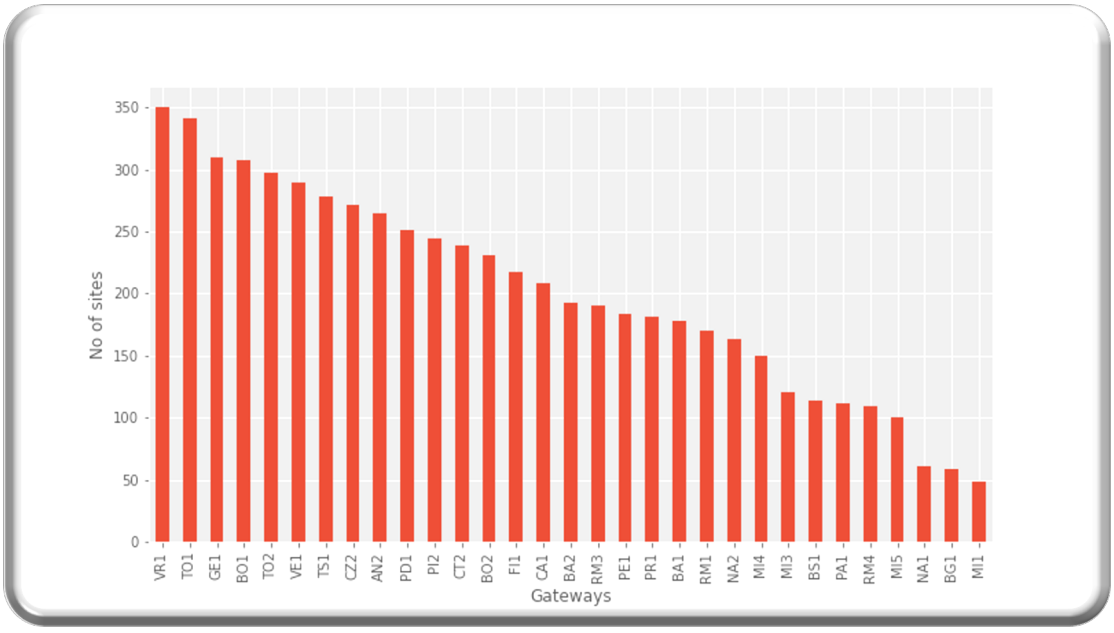

Figure 42: Bar chart showing the number of sites that fall above the set upload traffic volume threshold of 3000MB

With Download traffic Volume Threshold of 40000MB

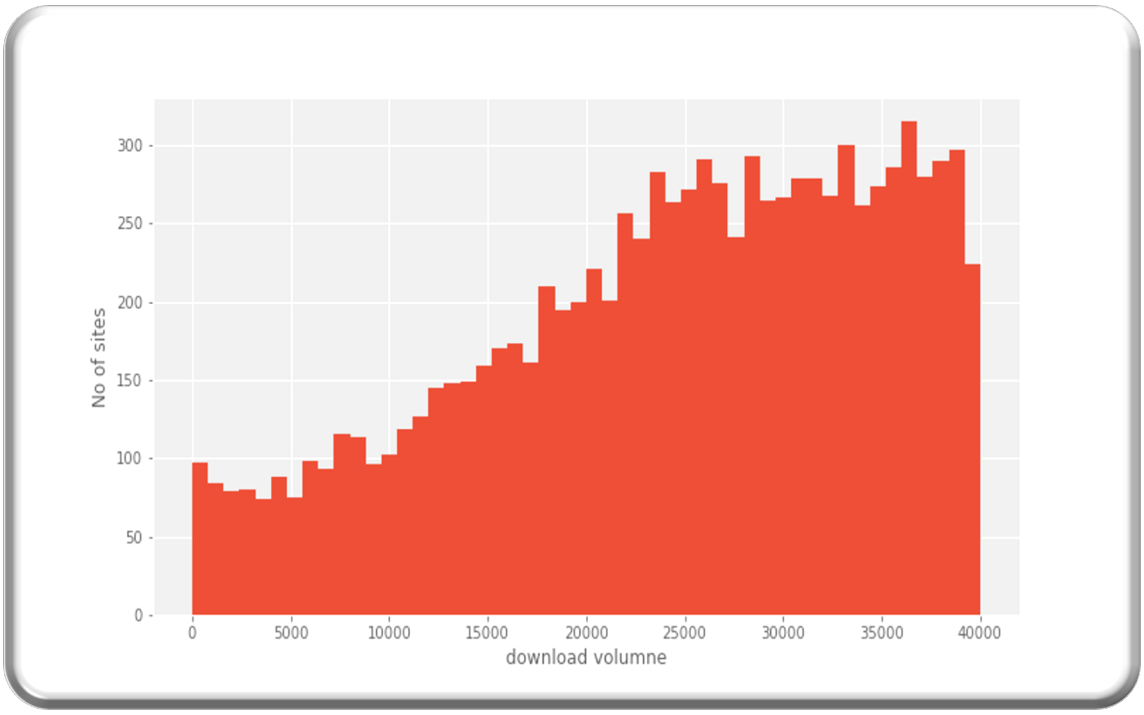

Figure 43: Histogram showing the number of sites that fall above the set download traffic volume threshold of 40000MB

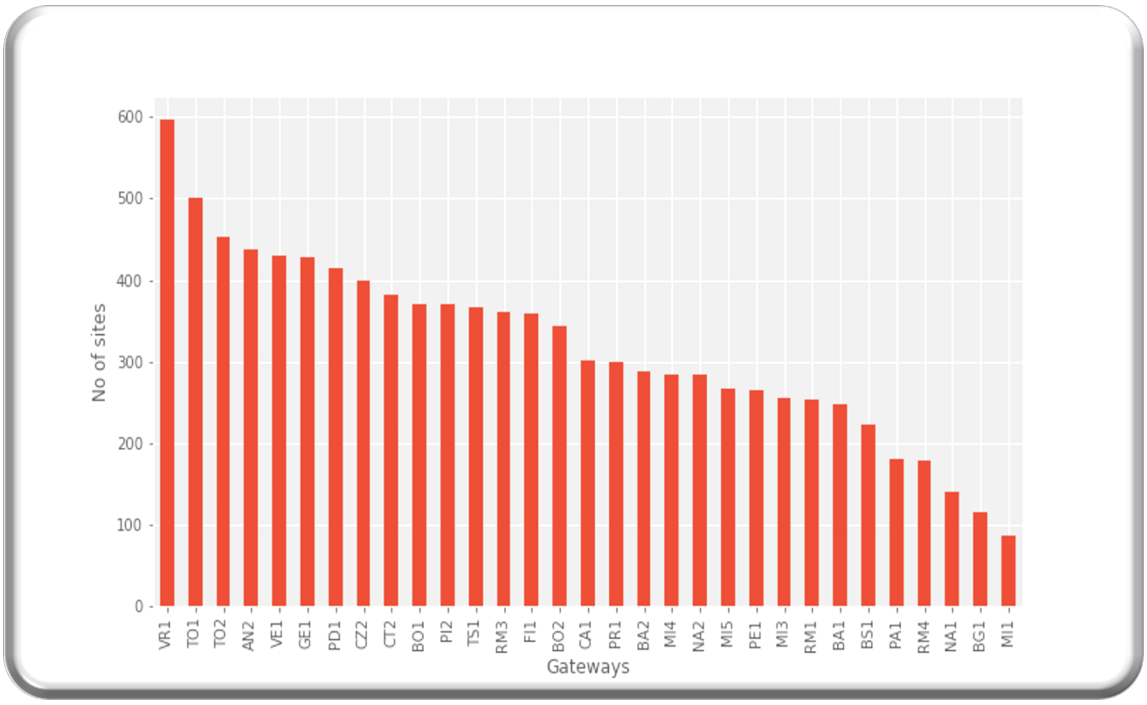

Figure 44: Bar chart showing the number of sites that fall above the set download traffic volume threshold of 40000MB

CHAPTER SIX

6 Conclusion

Mobile Edge Computing enables innovative service scenarios that can ensure enhanced personal experience and optimized network operation, as well as opening up new business opportunities. Mobile Edge Computing attracts a new value-chain and energized eco-system, where all players can benefit from tighter collaboration. Mobile operators can play a pivotal role within the new value chain and attract OTT service providers, developers and Internet players to innovate over a new cutting-edge technology, while enabling context-aware applications to run in close proximity to the mobile subscriber. Mobile subscribers can enjoy a unique, truly gratifying and personalized mobile-broadband experience which is tailored to their needs and preferences [25].

So far, this thesis has been able to focus on the Base5G project. It explained the Base5G reference architecture and the Base5G mathematical models for performing the techno-economic analysis. Numerical results applied to measurements collected on field show that, without MEC, URLL services can be deployed only to a small part of the network subscribers. On the other hand, if MEC could be enabled in all the candidate sites, then URLL services could be offered to almost all the network subscribers.

From the results of the histograms gotten from the different scenario, we have been able to confirm that MEC is effective to provide the lower delays required by URLLC applications. Also, the cost model was used to evaluate what the optimal cost MEC of nodes are and their optimal locations for different percentages of subscribers that need to be reached with URLL services. It can therefore be concluded that all of the objectives of this thesis work were met.

APPENDIX

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import random

#import data
detailed = pd.read_excel("RTT Data Poli v05.xlsx", sheet_name = 0)
detailed.head()

SecGw_red = pd.read_excel("RTT Data Poli v05.xlsx", sheet_name =
2).set_index("Unnamed: 0")
SecGw_red.head()
## clean the data for only sites with rtt values less than 40ms
detailed_cleaned = detailed[detailed['RTT [ms]'] <= 40]
detailed.shape
detailed_cleaned.shape

# SCENARIO A - All gateways are considered
def ScenarioA(data, thres,bin_size):
    valuesBelowThreshold = data[data['RTT [ms]'] < thres]
    plt.figure(figsize = (11, 6))
    plt.xlabel("RTT Value")
    plt.ylabel("No of sites")
```

```
    plt.hist(valuesBelowThreshold["RTT [ms]"], bins = bin_size)
    name = "ScenarioA-hist"+str(thres)+".png"
    plt.savefig(name)
    plt.show()

    #barchat
    plt.style.use('ggplot')
    plt.figure(figsize = (11, 6))
    plt.xlabel("Gateways")
    plt.ylabel("No of sites")
    data[data['RTT [ms]'] < thres]["SecGw SITE"].value_counts().plot(kind = 'bar')
    name = "ScenarioA-bar"+str(thres)+".png"
    plt.savefig(name)
    plt.show()
ScenarioA(detailed_cleaned,30,50)



# Scenario B
## calculating RTT with respect to MI01
respect = "MI1"
def ScenarioB(data,gateway_data,threshold,bin_size, respectTo):
    rtt_vals = list(data['RTT [ms]'])
    sites = list(data['SecGw SITE'])
    #get the gateway to gateway distances from MI01 to other gateways
    cols = gateway_data[[respectTo]]
```

```
row = dict(gateway_data.loc[respectTo, :])

# use the minimum gateway to gatway RTT and set nan values to 0
gw_RTT = []

for i in sites:
    gw_RTT.append(min([float(cols.loc[i]), row[i]]))

gw_RTT = [0 if math.isnan(i) else i for i in gw_RTT]

#add gateway distances, apply theshold and plot histogram
add = [sum(x) for x in zip(gw_RTT, rtt_vals)]

a = []
b = []

for i, j in zip(add, sites):
    if i < threshold:
        a.append(i)
        b.append(j)
plt.figure(figsize = (11, 6))
plt.xlabel("RTT Value")
plt.ylabel("No of sites")
plt.hist(a, bins = bin_size)
name = "ScenarioB-hist"+str(threshold)+".png"
```

```
    plt.savefig(name)
    #arrange the sites data
    site2MI01 = {"site2MI01": a}
    site2MI01_df = pd.DataFrame(site2MI01)
    site2MI01_df.index = b
    return (len(a),site2MI01_df)

ScenarioB(detailed_cleaned,SecGw_red,30,50,"MI1")



# Scenario C
#Cchoosing one Gateway to add to MI01
# function to get RTT values from  GATEWAY to a particular site
def gateway_dist(gatway_name,gateway_data,rtt,sites):
    cols = gateway_data[[gatway_name]]
    row = dict(gateway_data.loc[gatway_name, :])
    gw_dist = []

    for i in sites:
        gw_dist.append(min([float(cols.loc[i]), row[i]]))

    gw_dist = [0 if math.isnan(i) else i for i in gw_dist]
    add = [sum(x) for x in zip(gw_dist, rtt)]

    return add
```

```
# function return the gateway RTT matrix for all sites to all Gateways
def cal_gateway_dist(gateway_data,rtt,sites):
    all_gateway_calc = dict()
    for i in gateway_data.columns:
        values = gateway_dist(i,gateway_data,rtt,sites)
        all_gateway_calc[i] = values
    all_gateway_dist = pd.DataFrame(all_gateway_calc)
    all_gateway_dist.index = sites
    return all_gateway_dist
# function returns the gateway matrix when threshold is applied
def get_gateway_matrix(all_gateway_dists,gateway_data,threshold):
    gateway_matrix = pd.DataFrame()

    for i in gateway_data.columns:
        gateway_matrix[i] = np.where(all_gateway_dists[i] < threshold, 1, 0)
    return gateway_matrix
# sums the gateway matrix column wise
def get_gateway_matrix_sum(gateway_matrix,sites):
    #sum the values of the Gateway column wise to get the total number of sites below the
    #threshold
    new_row = dict(gateway_matrix.sum())

    for i, j in new_row.items():
        new_row[i] = [j]
```

```
    summed = pd.DataFrame(new_row)
    summed.index = ["SUM"]
    gateway_matrix_sum = gateway_matrix.append(summed)
    gateway_matrix_sum.index = sites + ["SUM"]
    return gateway_matrix_sum
# calculates all three matrices in one call
def Cal_AllGateWay_Matrix(gateway_data,data,threshold):
    rtt_vals = list(data['RTT [ms]'])
    sites = list(data['SecGw SITE'])
    all_gateway_dist = cal_gateway_dist(gateway_data,rtt_vals,sites)
    gateway_matrix = get_gateway_matrix(all_gateway_dist,gateway_data,threshold)
    gateway_sum_matrix = get_gateway_matrix_sum(gateway_matrix,sites)
    return (all_gateway_dist,gateway_matrix,gateway_sum_matrix)

def ScenarioC_Analysis_by_HighestNumber(gateway_matrix):
    ## Analysis 1, based on the highest number of sites covered
    ## highest number of sites covered
    temp = dict(gateway_matrix.drop(columns = ["MI1"]).sum())
    analysis_1 = dict()
    var = sorted(temp, key=temp.get, reverse = True)

    num = 1

    for i in var[:num]:
        analysis_1[i]=temp[i]
```

```
    return analysis_1
# returns the best coverage combination with MIO1
def ScenarioC_BestCoverage(iter,gatewaymatrix):
    Selected_Gateways = {}
    key = ""
    value = 0
    current = "MI1"
    standard = gatewaymatrix[current]
    step_data = gatewaymatrix.drop(columns=[current])

    #based on number of sites required loop through
    #and get the best combination to give the best coverage
    for j in range(iter):
        for i in step_data:
            temp = (standard | step_data[i]).sum()

            if temp < value:
                continue
            else:
                value = temp
                key = i

        current = key
        Selected_Gateways[key]=value
```

```
        standard = (standard | step_data[current])
        step_data = step_data.drop(columns=[current])

    return Selected_Gateways

### Execution of Scenario C starts here
#get the gateway matrix data by changing the accepted treshold
#returns three data items (gateway_RTT,  gateway_matrix,gateway sum matrix)
threshold = 30
swer = Cal_AllGateWay_Matrix(SecGw_red,detailed_cleaned,threshold)
gateway_dist = answer[0]
gateway_dist
gateway_matrix = answer[1]
gateway_matrix
gateway_matrix_sum = answer[2]
gateway_matrix_sum
analysis1 = ScenarioC_Analysis_by_HighestNumber(gateway_matrix)
analysis1
added_gateway = ScenarioC_BestCoverage(1,gateway_matrix)
added_gateway
MulitipleSelectedGatewayHist(added_gateway,gateway_dist,threshold,50)
## Execution ends

# Scenario D
## Multiple gateways selection
```

```
number = 4
gateways = ScenarioC_BestCoverage(number,gateway_matrix)
gateways
MulitipleSelectedGatewayHist(gateways,gateway_dist,threshold,50)



# Calculation based on traffic volume
def uploadVolume(data,threshold,bin_size):
    plt.figure(figsize = (11, 6))
    plt.style.use('ggplot')
    plt.xlabel("uploadload volumne")
    plt.ylabel("No of sites")
    plt.hist( data['UL_VOL [MB]'][data['UL_VOL [MB]']<= threshold], bins = bin_size)
    plt.savefig("UploadVolumeHist"+str(threshold)+".png")
    plt.show()


    plt.figure(figsize = (11, 6))
    plt.xlabel("Gateways")
    plt.ylabel("No of sites")
    data[data['UL_VOL [MB]'] <= threshold]["SecGw SITE"].value_counts().plot(kind = 'bar')
    plt.savefig("UploadVolumeBar"+str(threshold)+".png")
    plt.show()
def downloadVolume(data,threshold,bin_size):
    plt.figure(figsize = (11, 6))
    plt.style.use('ggplot')
```

```
    plt.xlabel("download volumne")
    plt.ylabel("No of sites")
    plt.hist( data['DL_VOL [MB]'][data['DL_VOL [MB]']<= threshold], bins = bin_size)
    plt.savefig("DownloadVolumeHist"+str(threshold)+".png")
    plt.show()

    plt.figure(figsize = (11, 6))
    plt.xlabel("Gateways")
    plt.ylabel("No of sites")
    data[data['DL_VOL [MB]'] <= threshold]["SecGw SITE"].value_counts().plot(kind = 'bar')
    plt.savefig("DownloadVolumeBar"+str(threshold)+".png")
    plt.show()
uploadVolume(detailed_cleaned,3000,50)
downloadVolume(detailed_cleaned,40000,50)



# Calculation based on cost values on each gateway
#randomly assign cost to the different gateways
gateways = SecGw_red.index
costs = {}
for i in gateways:
    costs[i]=random.randint(10,20)
print(costs)
```

```
## Getting Cost for selected gateways
def cal_cost_gateways(best_gateways, costs):
    sum = 0
    for i in best_gateways:
        sum+=costs[i]
    return sum
#calculating cost of selected gateways
number = 4
best_gateways = ScenarioC_BestCoverage(number,gateway_matrix)

#pass in best_gateways dictionary and weights dictionary
sum = cal_cost_gateways(best_gateways,costs)
sum

## Calculating Best Coverage Under Budget
#1. Calculate all possible combinations of gateways that
# the total sum of the cost is less than or equal to the budget

#2. based on the combinations found, calculate the coverage of each
#  Gateway combinations and select based on highest coverage and the lowest cost.
#takes in an array of dictionarys and coverts it to list
def convert_dictoList(array):
    val = []
    for i in range(len(array)):
        val.append(list(array[i])[0])
```

```
    return val
#recursive function takes in an array of dictionaries of gateway_cost
def combinations(gateway_cost, budget,lists,partial=[]):
    s = 0
    for i in range(len(partial)):
        s += list(partial[i].values())[0]

    if s > budget:
        total = s - list(partial[-1].values())[0]
        val = convert_dictoList(partial[:-1])
        val.append(total)
        lists.append(val)
        return lists

    for i in range(len(gateway_cost)):
        n = gateway_cost[i]
        remaining = gateway_cost[i+1:]
        combinations(remaining, budget,lists,partial + [n])
    return lists
def cal_allcombinations(gateway_cost,budget):
    l = []
    #convert the dictionary to arrays of dictionarys
    for i , j in gateway_cost.items():
        l.append({i:j})
```

```
    com = []
    #recursive function to check posible combinations
    com = combinations(l,budget,com)

    #remove duplicate values
    value = [list(item) for item in set(tuple(row) for row in com)]
    return value
def Cal_OptimumGatewayCombination(costs,budget,gatewayMatrix,fixed_gateway):
    #copy because dictionaries are not mutable, to create new
    #version of the gateway cost
    gateway_cost = costs.copy()
    #removing the cost of the fixed gateway
    budget = budget - costs[fixed_gateway]
    gateway_cost.pop(fixed_gateway)

    #calculate all combinations
    combinations = cal_allcombinations(gateway_cost,budget)

    coverage = 0
    cost = 0
    combination = []

    for i in range(len(combinations)):
        com = combinations[i]
```

```
        cols = [fixed_gateway] + com[:-1]
        cov_temp = (gatewayMatrix[cols].max(axis = 1)).sum()

        if(cov_temp < coverage):
            continue
        elif(cov_temp > coverage):
            combination = cols
            coverage = cov_temp
            cost = com[-1]
        else:
            if(cost > com[-1]):
                cost = com[-1]
                coverage = cov_temp
                combination = cols

    return ({"coverage":coverage},{"gateways":combination},{"Total
cost":(cost+costs[fixed_gateway])})
Budget = 50
optimized_value = Cal_OptimumGatewayCombination(costs,Budget,gateway_matrix,"MI1")
print(optimized_value)
```

REFERENCES

[1] Shaouha Wan, Sotirios K., Maode Ma, Houbing Song, and Shahid Mumtaz. “Mobile Edge Computing for 5G and Beyond: Emerging Trends and Applications” Hindawi Wireless Communication and Mobile Computing paper (2021).

[2] A. Craig, «Understanding augmented reality: concepts and applications,» Newnes, 2013.

[3] Elayoubi, Salah Eddine, et al, "5G RAN slicing for verticals: Enablers and challengers", IEEE *Communications Magazine 57.1,* pp. 28-34, (2019).

[4] Foukas, Xenofon, et al. "Network slicing in 5G: Survey and challenges." IEEE Communications Magazine 55.5 (2017): 94-100

[5] Ordonez-Lucena, Jose, et al. "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges." IEEE Communications Magazine 55.5 (2017): 80-87

[6] Han, Bin, Shreya Tayade, and Hans D. Schotten. "Modeling profit of sliced 5G networks for advanced network resource management and slice implementation." 2017 IEEE symposium on computers and communications (ISCC). IEEE, 2017

[7] 5G Americas, «Network Slicing for 5G and Beyond,» White Paper, 2016.

[8] Kekki, Sami, et al. "MEC in 5G networks." ETSI white paper 28 (2018): 1-28.

[9] N. Abbas, "Mobile Edge Computing: A Survey", Unpublished Master's thesis, University of Oslo, 2016.

[10] M. T. B. e. al, "Mobile edge computing: A taxonomy," in *proc. of the sixth International conference on Advances in Future Internet*, Citeseer, 2014.

[11] Q.-V. P. e. al, "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," *IEEE*

COMMUNICATIONS SURVEYS AND TUTORIALS, p. 5, 2019.

[12] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017

[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," IEEE Communications Surveys Tutorials, vol. 19, no. 3, pp. 1657–1681, Third Quarter 2017.

[14] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Communications Surveys Tutorials, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017

[15]] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," IEEE Access, vol. 5, pp. 6757–6779, Mar. 2017

[16] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," IEEE Communications Surveys Tutorials, vol. 20, no. 1, pp. 7–38, First Quarter 2018.

[17] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," IEEE Access, vol. 6, pp. 3962–3976, Jan. 2018.

[18] J. M. a. D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Communications Surveys Tutorials,* vol. 21, n. 1 , p. 260–288, First Quarter 2019.

[19] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[20] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," IEEE Systems Journal, vol. 12, no. 3, pp. 2495–2508, (Sep. 2018).

[21] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," IEEE Communications Surveys Tutorials, vol. 20, no. 4, pp. 2961–2991, Fourth Quarter 2018.

[22] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," IEEE Access, vol. 6, pp. 6900–6919, (Mar. 2018).

[23] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," Future Generation Computer Systems, vol. 78, pp. 680 – 698, (Feb. 2018)

[24] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2586–2595, (Nov. 2017).

[25] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher and Valerie Young, " Mobile Edge Computing A key technology towards 5G," ETSI White Paper, no. 11, (Sep 2015).