Predictive Admission Control and Bandwidth Allocation Scheme for Integrated Terrestrial and Non-Terrestrial Network.

Prepared by:
Kananelo Chabeli

Prepared for:
A/Prof Olabisi E Falowo
Department of Electrical Engineering
University of Cape Town

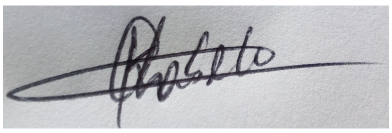
Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering.

December 6, 2024

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed and has been cited and referenced. Any section taken from an internet source has been referenced to that source.
3. This report is my own work and is in my own words (except where I have attributed it to others)
4. I have not paid a third party to complete my work on my behalf. My use of artificial intelligence software has been limited to checking for grammatical errors in text and scoring the clarity of my text.
5. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.
6. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Word count: 19093



December 6, 2024

Kananelo Chabeli

Date

Terms of Reference

EEE4022S/F Topic template

Student proposed?	*Y/N* *N*	*If Y, student name*
ID:	**OF24-06**	
SUPERVISOR:	Olabisi Falowo	
TITLE:	**Predictive Admission Control and Bandwidth Allocation Scheme for Integrated Terrestrial and Non-Terrestrial Network**	
DESCRIPTION:	The sixth generation (6G) mobile network will combine the terrestrial and non-terrestrial networks to provide ubiquitous coverage and consistent QoS to different groups of users in a flexible manner. Thus, a user may be connected through different types of networks during a communication session. Incorporating a predictive technique in admission control and allocation of bandwidth for diverse users' services can enhance QoS provisioning and radio resource utilization efficiency in the 6G network. The purpose of this project is to review existing call admission control and bandwidth allocation algorithms and develop a predictive call admission control and bandwidth allocation scheme for the 6G network. An aspect that may be exploited in this project is the prediction of individual user's service time.	
DELIVERABLES:	Literature review, predictive call admission control and bandwidth allocation scheme, simulation results, simulation code, and report.	
SKILLS/REQUIREMENTS:	MATLAB, Python, or any other programming language, Knowledge of EEE4121F.	
GA 1: Problem solving: *Identify, formulate, analyse and solve complex* engineering problems creatively and innovatively*	The student is expected to (1) review existing call admission control and bandwidth allocation algorithm, (2) design a predictive call admission control and bandwidth allocation algorithm, and (3) implement the call admission control and bandwidth allocation algorithm.	
GA 4: Investigations, experiments and analysis:** *Demonstrate competence to design and conduct investigations and experiments.*	The student is expected to investigate the performance of the predictive call admission control and bandwidth allocation algorithm through simulations.	
GA 5: Use of engineering tools: *Demonstrate competence to create, select and apply and recognise limitations of appropriate techniques, resources and modern engineering and IT tools, including prediction and modelling, to complex engineering problems*	The student is expected to develop a network model, develop a predictive call admission control and bandwidth allocation algorithm, and implement the algorithm using MATLAB or any other programming language.	
EXTRA INFORMATION:	For a student interested in pursuing a master's degree, the project can be expanded to an MSc dissertation.	
BROAD Research Area:	Wireless Networks	
Project suitable for ME/ ECE/EE/ALL?	EE/ECE students who have taken EEE4121F course.	

12

Acknowledgements

I would like to first thank God, Almighty, for the wisdom He has given in completing this work. I would like to thank the Government of Lesotho for funding my studies at the University of Cape Town; without them, this invaluable milestone could not have been reached.

Next, I would like to thank my supervisor, A.Prof. Olabisi.E. Falowo, for his support, guidance, and insightful suggestions throughout the duration of this project. His invaluable feedback has been helpful in shaping the direction of this project.

A special thanks to my family for the unwavering support they have given me throughout this project. To my mother, I would like to send special thanks to you for always believing in me, not only in academics, for the broader spectrum of aspects of this life. Also, I would like to appreciate the encouragement that my wife, 'Makhalalelo Chabeli, has given me in this project. Her prayers and strong belief in me went a long way in making this work successful.

I would like to dedicate this work to my son, whose presence has brought immense joy and meaning to my life. You are a constant source of inspiration, and your smile reminds me every day of what truly matters. This work is for you, and I hope that one day it will inspire you to pursue your own passions and dreams.

Abstract

The advancements in the field of telecommunications have catapulted the world into the fourth industrial revolution. With each generation making a leap of a decade, the current generation of wireless networks being deployed world-wide is the fifth-generation (5G). Although 5G shows significant advancements in establishing connected society with high data rates and capacity, it may fall short for meeting stringent requirements of emerging applications and demands of 2030 and beyond. To address this, the next-generation network (6G) has gained considerable momentum in research community.

It has been envisioned that 6G will achieve global coverage by integrating terrestrial and non-terrestrial networks. However, this integration presents a number of challenges as a result of movement of non-terrestrial networks such as Low-Earth Orbit(LEO) satellites, and High Altitude Platforms(HAPs). Although there has been a significant advancements in development of efficient call admission control for efficient resource allocation in heterogeneous wireless networks, few have been developed to address the challenges that arise from integrating terrestrial and non-terrestrial networks.

This paper proposes a predictive call admission control and bandwidth allocation scheme to reduce the frequency hand-offs in integrated terrestrial and non-terrestrial networks. The proposed scheme leverages on Generalised Linear Model(GLM) to predict service duration during call request, and factor in this duration in admission decision making. With this predictive scheme, this paper seeks to underscore the importance of considering duration of services in resource allocation. The simulation results show that, compared to admission control that does not consider service duration, the proposed scheme achieves significantly low satellite handoff rates when service duration is considered in admission control.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

For the past several decades, telecommunications have experienced remarkable technological advancements [1]. From the first-generation (1G) analogue systems to fifth-generation (5G), the evolution of wireless cellular communications has brought significant improvements in multiple access techniques, spectral efficiency, and data rates. Current 5G networks are making substantial progress toward achieving a fully connected society, introducing paradigm shifts such as network virtualisation, software-defined radio, and network slicing [2]. With capabilities like 20 Gbps peak data rates, 1 ms end-to-end delay, and 100 times spectral efficiency [3], 5G has transitioned from merely connecting humans to enabling machines and 'things' to transmit data at high speeds.

Despite their remarkable capabilities, 5G networks may struggle to meet the stringent demands of the 2030 society and beyond due to increasing user requirements [3]. This has driven the research community to shift its focus toward sixth-generation (6G) networks. One of 6G's key visions is to achieve near 100% global coverage, enabled by the integration of terrestrial and non-terrestrial networks (NTNs), such as satellite communication systems. Low Earth Orbit (LEO) satellites play a crucial role in providing global coverage and ubiquitous connectivity [4], thanks to their low altitudes, which result in reduced propagation delays and an improved radio link budget. As a result, the current 3rd Generation Partnership Project (3GPP) standardisation focuses on incorporating NTNs into the 5G ecosystem

First studied by 3GPP in Release 15 [5], the integration of non-terrestrial networks (NTN) within 5G remains an area of extensive research and development. 3GPP Release 17 marks the first standardisation to incorporate satellite technology into the 5G New Radio (NR) standard [5, 4]. This has spurred research projects like the European Space Agency (ESA)-sponsored 5G-GOA (enabled ground segment technologies over-the-air demonstrator) [6] and 5G-LEO (OpenAirInterfaceTM extension for 5G satellite links) [7], aimed at extending the OpenAirInterface (OAI) framework to develop simulators and in-lab demonstrators for 5G-NTN in compliance with Release 17.

1.2 Objectives of the Study

1.2.1 Problems to be Investigated

The problem being investigated in this study is the high frequency of Low Earth Orbit satellite handoffs in integrated terrestrial and non-terrestrial networks. Because of their low orbital altitudes, LEO satellites move at high speeds and have small coverage areas, leading to frequent handoffs. This increases the risk of handoff failures [4], which can disrupt communication and degrade network performance.

1.2.2 Purpose of the Study

This study proposes a predictive admission control and bandwidth allocation scheme to reduce frequency of handoffs in integrated terrestrial and non-terrestrial networks. The proposed scheme leverages on Generalised Linear Model to predict service duration and factor this in admission control decisions. Thus, the purpose of this study is to underscore the significance of considering service duration in admission control decision in integrated terrestrial and non-terrestrial networks.

1.3 Scope & Limitations

This study leverages on generalised linear Model to predict service duration. The algorithm is trained and evaluated on synthetic call detail record data. This study is limited to only software implementation of the proposed scheme with no hardware or real-time network implementation. Furthermore, the study evaluates the performance of the proposed predictive admission control using only call blocking probability and satellite handoff probability.

1.4 Plan of Development

The rest of this paper is organized as follows: In Chapter 2 we provide detailed overview of the evolution path of wireless cellular networks followed by review of current state-of-the-art literature in admission control and bandwidth allocation schemes. Moving on, Chapter 3 presents the system model of the propose admission and bandwidth scheme. Next, Chapter 4 details the implementation of this scheme, describing how simulation data was gathered and analysed. In Chapter 5, we present the simulation results and the analysis of the proposed scheme. Chapter 6 draws conclusions based on the results and based on these conclusions, recommendations for future work are made.

Chapter 2

Literature Review

2.1 Introduction to Literature

The advent of new technologies and increasingly stringent user demands have driven the continuous development of mobile and wireless networks. These networks are globally standardized by the International Telecommunication Union (ITU) and the 3rd Generation Partnership Project (3GPP). The ITU defines the overall framework of cellular technology, including technical and performance specifications, while the 3GPP produces reports and standards for 3GPP mobile technologies.

This chapter presents overview of evolution path of wireless cellular networks from first generation(1G) analog systems to current fifth generation(5G) networks. Next, a detailed overview of the next generation wireless networks is presented followed by review of current works in admission control and bandwidth allocation schemes, particularly focusing on predictive bandwidth allocation schemes.

2.2 Evolution of Cellular Networks.

2.2.1 The Cellular Concept

Advancements in telecommunications and microelectronics have ushered the world into the fourth industrial revolution. Over the decades, mobile technology has evolved through five generations, beginning with 1G in the 1980s and progressing to the current deployment of 5G worldwide. To fully appreciate the profound impact of these advancements, it is essential to first understand the foundational concept of cellular technology.

Pre-cellular systems known as mobile communication systems, were designed specifically for a few mobile users. These systems involved the installation of large conventional base station(BS) that covered wide geographical area, requiring excessive transmission power [8]. As a result, The systems had the disadvantage of limited mobility because hand-offs were not possible. Additionally, all of the allocated spectrum was used by a single base station, resulting in limited system capacity.

The concept of cellular network was first presented in United States of America(USA) by engineers at Bell Laboratories in 1947 [8]. In cellular networks, a geographical area is divided into hexagonal regions called cells. Each cell consists of a base station , at its centre and a group of base stations is controlled by Mobile Switching Centre(MSC). Moreover, every base station in the network is assigned a portion of the allocated spectrum,in such a way that no adjacent cells used same channels(avoiding co-channel interference). With this configuration, the system has high capacity because frequency channels can be re-used among co-channel cells.

2.2.2 First Generation Networks (1G)

1G is the generation of cellular mobile networks that first realized the cellular network concept proposed in 1947 by Young[8]. Despite its proposal in 1947, the initial design of a cellular system began in the 1960s due to hardware and technological barriers. The first commercial deployment of a cellular network was in Japan in 1979, even though its origin was in the USA [8].

However, there is a divergence in the literature regarding when and where 1G was first launched. For instance, the authors in [9, 10, 11] date the origin of 1G to 1980, while [12] suggests the birth date as early as 1979 and as late as 1982 according to [13]. This inconsistency is due to the lack of an internationally agreed-upon 1G standard. Each 1G standard operated within a confined region or country, resulting in inefficient use of spectrum, especially at borders and no global roaming was supported. In North America, the 1G standard was called the Analog Mobile Phone System (AMPS) and was allocated the frequency band of 800-900 MHz [13]. Meanwhile, in Europe and Asia, it was known as the Total Access Communication System (TACS) [9], and in Germany and South Africa, it was referred to as C-Netz [8].

All 1G standards were designed for analog transmission of voice and were based on circuit-switching technology. Circuit-switching is a technology where an end-to-end connection must be established between communicating nodes before communication can begin. The purpose of establishing this connection is to reserve enough radio resources for the connection (circuit) or decline the call if there are insufficient resources available. This approach had the benefit of completely avoiding congestion and ensuring guaranteed end-to-end quality of service. However, it resulted in wasted resources because they could not be reallocated when the connection was idle.

Features of 1G include transmission data rate of 2.4 Kbps [14, 12], and traffic was multiplexed using the Frequency Division Multiple Access (FDMA) technique [9, 13, 14]. Key enabling technologies are cell splitting, cell sectoring, and handover support. As the number of subscribers increases, the system capacity becomes limited. Cell splitting is a technique introduced in 1G to increase system capacity. This involves splitting a congested cell into multiple smaller cells, each with its own base station [8]. This approach increases the number of times a frequency channel is reused, effectively increasing system capacity. This differs from cell sectorization is that, instead of splitting a cell into multiple smaller cells, a single cell is divided into sector (typically 120°) where each sector is served by an antenna. Channels allocated for that cell,are divided among the sectors.

Although 1G showed significant breakthrough in 1G networks, it faced a number of challenges. For instance, 1G supported only voice communication [12],and the use of analog signal presented security concern as digital encryption could not be applied to them [14]. Additionally, 1G had limited capacity with extremely inefficient use of spectrum. 1G had poor signal reception [15] and was subject noise resulting a unclear voice communication. This limitations made 1G to evolve into 2G.

2.2.3 Second Generation Networks (2G)

The second-generation (2G) standards are the world's first digital cellular technologies, and the formulation of an international mobile communication standard first commenced with 2G networks [12]. 2G standards were introduced in the late 1980s [16, 17] to solve the limitations of 1G cellular networks. They were deployed worldwide in the early 1990s [12, 13]. The common 2G standards are IS-95, IS-136, and the Global System for Mobile Communications (GSM), with GSM being the most popular, first deployed in 1991.

2G cellular networks use digital techniques such as Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA), and are designed for digital voice transmission as well as low-speed Short Message Service (SMS), a data service. Furthermore, 2G networks support data rates up to 64 Kbps and provide clear voice calls [14]. 2G achieved improved security through digital encryption.

As a result of increasing demands for data transmission over the air interface, GPRS, a 2.5G standard technology, was introduced in the mid-1990s [14] into the existing GSM [9]. GPRS is a packet-switched technology that can reach data rates up to 115 Kbps [10]. The introduction of GPRS involved the addition of interfaces such as SGSN (Serving GPRS Support Node) and GGSN (Gateway GPRS Support Node) into the existing GSM network. Furthermore, the transmission of data over the air interface gave rise to internet services, and packet-switching introduced the Internet Protocol (IP).

However, GPRS still couldn't handle increasing interest in the internet and the demands for high data rates. This resulted in GPRS evolving into EDGE. The Enhanced Data rates for GSM Evolution (EDGE) was the next phase in the evolution path of 2G. It is commonly known as the 2.75G standard and paved the way for GSM to support enhanced data rates [12]. For instance, typical rates of 200 Kbps [12] and up to 500 Kbps [11] could be reached. Furthermore, EDGE was designed to enhance the packet-switching services of GPRS and support future applications requiring high-speed data. It is the technology that gave rise to the increasing interest in today's internet services.

2.2.4 Third Generation Networks(3G)

In EDGE, high data rates and large volumes of data movement were possible. However, the packet transfer air-interface still behaved like a circuit-switched call [13], leading to inefficiencies. Furthermore, with 2G, the standards for network development varied across different regions of the world. As a result, the 3GPP organization was formed to assist in the development of 3G, which aimed to provide services globally, independent of the underlying technology. 3G was first launched in Japan in 2001 [13, 18], known as the Universal Mobile Telecommunication System (UMTS) in Europe, while the American variant was called CDMA2000.

The key features of the 3G cellular network include high data rates of up to 2 Mbps and support for diverse classes of services. With 3G, television (TV) services could be streamed on mobile phones, and live video streaming became possible. It also offered the ability to download TV streams for offline viewing. Furthermore, 3G supported high mobility of up to 100 km/h [9] with efficient handover. As [18] notes, 3G merges technologies such as TDMA, CDMA, and GSM to achieve high spectral efficiency and backward compatibility. It can be considered the bridging point between packet-switched and circuit-switched networks, as it supported both technologies, whereas 4G is entirely packet-switched.

The key enabling technologies of the 3G wireless cellular network include Wide-band Code Division Multiple Access (WCDMA), which was launched by the GSM community in 2014 from Finland [18], and High-Speed Packet Access (HSPA) [12]. WCDMA was also used for mobile data transmission via satellites at high speeds. HSPA combines two technologies: High-Speed Up-link Packet Access (HSUPA) and High-Speed Down-link Packet Access (HSDPA), providing better end-to-end network performance [12]. As a result of these improvements and the increasing interest in the internet, the number of 3G subscribers increased exponentially, presenting several challenges to the network. For instance, it became necessary to obtain new spectrum, but licensing spectrum from government authorities proved difficult [18]. Additionally, increased internet usage introduced new security challenges, and the growing demand for high data rates and reliability motivated the evolution of 3G into the 4th Generation (4G).

2.2.5 Fourth Generation Network(4G)

The fourth generation of cellular technology, commonly known as 4G, was first field-tested by NTT in Japan in June 2005 [13]. However, Long-Term Evolution (LTE), a 3GPP 4G standard, was not commercially launched until 2010 in Finland [18]. This technology is often referred to as 'ALL-IP' because all traffic is carried over a packet-switched evolved packet core (EPC). 4G was designed with the concept of connecting 'everything everywhere,' effectively promoting the emergence of massive Internet of Things (IoT) applications. It operates as a heterogeneous network where different radio access technologies coexist within the same geographical area.

In addition, the 4G networks were engineered to support seamless mobility, quality of service (QoS), and minimal latency [19], with data rates of up to 100 Mbps for mobile environments and 1 Gbps for indoor environments [17]. This technology enables streaming of high-resolution videos and promotes machine-type communications (MTC). Key enabling technologies for 4G include network densification, carrier aggregation [12], orthogonal frequency division multiple access (OFDMA) for downlink and single carrier FMDA for the uplink[18, 19, 20], and multiple-input multiple-output (MIMO) systems.

Network densification is particularly useful in densely populated urban areas, where small, low-cost, and low-power cells are deployed to provide low coverage per cell, thereby increasing capacity and data rates. As noted by [19], the major enhancements of 4G are in multi-casting and interference mitigation. OFDMA allows the division of data symbols among orthogonal narrow-band sub-carriers [18], reducing transmission rates, resulting in longer symbol duration, and making the system more robust against inter-symbol interference (ISI).

To achieve high spectral efficiency, 4G employs MIMO technology and carrier aggregation in addition to OFDMA. Carrier aggregation involves combining several carriers in the physical layer to achieve the required high data rates. MIMO, on the other hand, is a technique that enables the simultaneous transmission and reception of data using multiple antennas.

While 4G technology brought significant advancements over its predecessors, 3G and 2G, it had several limitations. For instance, it was primarily deployed in urban areas, leaving remote rural regions with little coverage. This created a digital divide in society, where urban users had access to more advanced internet technologies than their rural counterparts. Additionally, the emergence of virtual reality (VR) and augmented reality (AR) applications required extremely low latency, beyond what 4G could offer. Although 4G supported IoT devices, it was not significantly optimized for the massive scalability of IoT technologies in the future. These limitations highlighted the need for further advancements, leading to the development of the 5G standard.

2.2.6 Fifth Generation Networks(5G)

The deployment of 5G wireless cellular systems started in 2020 [21]. Thus, it is currently in its new stage of deployment. The emergence of new applications and stringent user requirements have triggered the evolution of wireless cellular systems [22], and the primary goal of 5G systems is to overcome the limitations of 4G systems. Contrary to other generations, 5G is user-centric, optimized to prioritize user requirements as opposed to being operator-centric and service-centric [23]. This has made 5G more reliable and to have higher capacity.

Furthermore, 5G is more flexible because it allows diverse deployments. It can be deployed as a stand-alone(SA) or non-stand alone(NSA). Its main characteristics include high data speeds(upto 20Gbps), low latency(1ms), and high capacity [23]. It uses efficient orthogonal frequency division multiple Access (OFDMA), which significantly improves spectrum efficiency. New features added in the 5G standard include (but are not limited to) software-defined networking (SDN), network slicing, network function virtualization (NFV), and massive multiple input multiple output (MIMO) communication technology.

The Visions and Use Cases of 5G Networks

The 5G network is expected to support enhanced Mobile Broadband(eMBB),ultra-Reliable Low Latency Communications (uRLLC) and massive Machine-Type Communications (mMTC) slices[21, 24, 25]. eMBB slice features mobile network services that transmit large volumes of data and extremely high speed. It requires multi-Gbps data rates and extremely high capacity. On the other hand, uRLLC includes services such as e-health, robotics,and automation which require extremely high reliability and low latency with strong security. In addition, mMTC, presents additional requirement to 5G standard by requiring high density network, deep coverage network and ultra-low energy. These diverse service requirements can only be fulfilled by allocating dedicated network slices [25].

The Key Requirements of 5G Networks

For 5G networks to meet, eMBB data rates requirements, it supports peak data rates of upto 20 Gbps. The reliability requirement of uRLLC is 99.9999% [26] and 5G meets this by having a high reliability of 99.9999% [3]. Moreover, the high density of 1 million devices per km^2 makes 5G suitable for mMTC applications. 5G also features an number of enabling technologies such as SDN, NVF and Network Slicing.

Enabling Technologies for 5G Networks

Software Defined Networking(SDN) is a technology that offer capability of controlling traffic remotely important not only to Network operators,but also communication network research community. Separation of control and data planes presents a number of benefits. First, it makes it easy to control and monitor the network traffic. Second, SDN makes is possible for software (control plane) to evolve independent of hardware(data plane). Third, separating control and data plane improved network security because it is easier to detect intrusion into network. Lastly, SDN paves the way for Network Slicing and NFV.

Network Slicing is an important feature that was introduced in 5G that enables creation of logically independent network slices on the same physical infrastructure [25, 27]. Each slice can be tailored to provide specific user requirements. Thus, with Network slicing, eMBB, uRLLC and mMTC slices can co-exist on the same physical infrastructure. Another important feature of 5G technology is **Network Function Virtualization**(NFV). It is a functionality that packages network functions so that they can commonly be run of the same physical hardware[25].

Limitations of 5G Networks

5G network being a terrestrial network faces a number of limitations. First, terrestrial networks are subject to natural disasters such as earth-quakes,and a unable to handle surge network demands [28]. Society in 2030 and beyond will require processing of large volumes of data near real-time needing extremely high throughput and ultra-low latency, beyond what 5G can offer. Further more introduction of flying cars, extended reality and telemedicine requires extremely high data rates of multi Tbps which 5G can not offer [21]. As a result, the research community are showing much interest in the upcoming 6G technologies. 6G networks are reviewed in the next section.

2.3 The Next Generation Networks: Towards 6G

2.3.1 Introduction to 6G Networks

In response to challenges faced by 5G networks, the sixth-generation (6G) wireless networks have gained considerable momentum in research on wireless communication networks [29]. Notably, Finland led the way by organizing the first 6G summit and initiating the world's first 6G project, known as 6Genesis [30]. Other notable 6G projects include China's 'Broadband Communication and New Networks' and several beyond-5G (B5G/6G) initiatives sponsored by the European Commission's Horizon 2020 program, including the well-known Terranova project [30].

The sixth generation of wireless networks is envisioned as the full realization of the fourth industrial revolution. It will be a hybrid network that integrates both terrestrial and non-terrestrial networks to connect everything, everywhere. Furthermore, 6G is anticipated to harness the complete potential of Artificial Intelligence (AI) and Machine Learning (ML) to drive a truly intelligent society. This section provides an overview of the state-of-the-art work and paradigm shifts in wireless communication technologies.

2.3.2 The Vision of 6G Networks

Although the fifth generation (5G) of wireless networks offers significant advances beyond LTE, it may not be sufficient to meet the future demands of the digital society [31]. This makes the launch of the sixth generation (6G) inevitable [32]. While 5G extends the Internet of Things (IoT) paradigm of 4G into the Internet of Everything (IoE) [33], 6G will further enhance the IoE by deeply integrating AI and ML into the network. Additionally, 6G will integrate both terrestrial and non-terrestrial networks to provide full global coverage, ensuring seamless connectivity for everything, everywhere [3].

The sixth generation is envisioned to go beyond just communication by incorporating sensing, communication, positioning, advanced radar, and navigation capabilities into the network [30]. 6G will be both human- and machine-centric, providing multiple ways for figures, voices, and eyes to communicate. It will be a multi-band and hyper-flexible communication system that integrates various vertical communication technologies [32]. Figure 2.1 provides an overview of the 6G vision. The next subsection explores the potential usage scenarios of 6G.

2.3.3 The Usage Scenarios of 6G Networks

In 2015, ITU-R M.208.3 [34] identified three key usage scenarios for 5G: enhanced Mobile Broadband (eMBB), ultra-Reliable Low Latency Communication (uRLLC), and massive Machine-Type Communication (mMTC). eMBB was designed to address human-centric applications with extremely high data rates, while uRLLC was intended to support mission-critical applications such as autonomous vehicles or remote surgery. mMTC, on the other hand, was envisioned to support diverse connectivity with a large number of IoT devices [35].

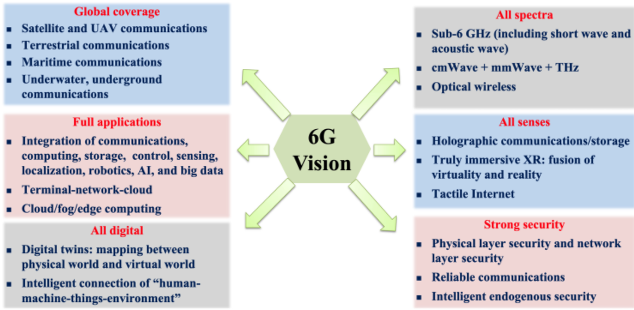


Figure 2.1: The vision of 6G [33].

The sixth generation networks are expected to extend these scenarios into further enhanced Mobile Broadband (feMBB), extreme ultra-Reliable Low Latency Communication (euRLLC), and ultra-massive Machine-Type Communication (umMTC) [3]. Beyond extending these existing scenarios, 6G will also introduce three new usage scenarios: ubiquitous Mobile ultra-Broadband (uMUB), ultra-high-speed-with-low-latency communication (uHSLLC), and ultra-high data density (uHDD) [21].

Furthermore, 6G is expected to unify various 5G use cases into new scenarios to support diverse vertical applications. For example, Rasti el al. [32] envision that 6G will integrate eMBB and uRLLC into Mobile Broadband Reliable Low Latency Communications (MBBRLLC), supporting both high broadband data rates and high reliability with low latency. In contrast, Bafanaa et al. [36] argue that eMBB and uRLLC will be unified into ultra-Reliable Low Latency Broadband Communications (uRLLBBC) to support applications that require not only high reliability and low latency but also extremely high throughput. Additionally, Banafaa et al. [36] foresee that 6G will integrate mMTC and uRLLC into massive ultra-Reliable Low Latency Communication (mULLC) to support a large number of actuators and sensors with stringent requirements on latency and reliability.

In summary, 6G will not only extend the existing 5G usage scenarios but will also introduce new use cases and unify different 5G use cases into new scenarios to support diverse vertical applications. The next subsection highlights the 6G performance indicators provided in the literature to support these usage scenarios.

2.3.4 Key Performance Indicators(KPIs) for 6G

To fulfill the visions and usage scenarios outlined for 6G networks, the current state-of-the-art works highlight several key performance indicators (KPIs) that will define the capabilities of 6G:

- **High Peak Data Rates:** While 5G can achieve peak data rates of up to 20 Gbps[3], 6G is expected to reach peak rates ranging from 1 to 10 Tbps[3, 30, 32]. These extraordinary data rates will support data-intensive applications, such as holographic-type communications (HTC), which require up to 4.32 Tbps[31], as well as THz wireless backhaul and fronthaul[30].
- **User-Experienced Data Rate:** Applications such as Augmented Reality (AR) and Virtual Reality (VR) cannot be effectively compressed for real-time interactive environments. Therefore, 6G will need to provide user-experienced data rates of up to 1 Gbps, significantly higher than the 100 Mbps offered by 5G.
- **Latency:** To support mission-critical applications, such as those under the extreme ultra-Reliable Low Latency Communication (euRLLC) scenario, 6G will need to achieve extremely low latency, with over-the-air latency of 10-100 μs and end-to-end latency of 10 ms.

A comprehensive set of 6G KPIs, including comparisons with 4G and 5G, is provided in Table 2 of [36] and is illustrated in Figure 2.2. The next section explores the key enabling technologies that will support these KPIs.

KPIs	4G	5G	6G
Peak data rate /device	1 Gbps	10 Gbps	1 Tbps
latency	100 ms	1 ms	0.1 ms
Max. spectral efficiency	15 bps/Hz	30 bps/Hz	100 bps/Hz
Energy efficiency	< 1000x relative to 5G	1000x relative to 4G	>10x relative to 5G
Connection density	2000 devices / km^2	1millon devices /km^2	>10millon devices/km^2
Coverage percent	< 70 %	80 %	>99 %
Positioning precision	Meters precision (50 m)	Meters precision (20 m)	Centimeter precision
End-to-end reliability	99.9 %	99.999 %	99.9999 %
Receiver sensitivity	Around −100dBm	Around −120dBm	< −130dBm
Mobility support	350 km/h	500 km/h	≥1000 km/h
Satellite integration	No	No	Fully
AI	No	Partial	Fully
Autonomous vehicle	No	Partial	Fully
Extended Reality	No	Partial	Fully
Haptic Communication	No	Partial	Fully
THz communication	No	limited	Widely
Service level	Video	VR, AR	Tactile
Architecture	MIMO	Massive MIMO	Intelligent surface
Max. frequency	6 GHz	90 GHz	10 THz

Figure 2.2: Comparison of Requirements between 4G, 5G, and 6G [36]

2.3.5 Key Enabling Technologies(KETs) for 6G

The sixth generation (6G) networks will require extremely high data rates and capacity. While the introduction of mm-Wave technology in the 5G era showed great potential for increasing system capacity, it will not be sufficient to support the density of 10^7 devices per square kilometer required by 6G. Therefore, 6G will need to utilize the Terahertz (THz) band to support multi-Tbps data rates. For instance, the authors in [32, 30] note that the 0.1-10 THz band will be licensed, while the 400-800 THz band will be unlicensed, supporting applications such as 3D holographic communications and pervasive connectivity. Additionally, 6G will employ ultra-massive spatially modulated MIMO (UM-SM-MIMO) to achieve the required capacity [32]. Building on the evolution from 4G's eight-antenna MIMO to 5G's 256-1024 antenna MIMO, 6G is expected to deploy up to 10,000 antennas.

To provide global coverage and seamless connectivity, 6G will integrate space (satellite), air (AUV-assisted networks), underground, and terrestrial networks. It will harness the power of Artificial Intelligence (AI) and Machine Learning (ML) to optimize network performance [21]. Furthermore, blockchain technology will be explored for efficient radio management and spectrum sharing [30]. Edge computing technology will also be scaled in the 6G era to connect computing devices to the network.Additionally, security in 6G will be significantly enhanced through the use of quantum computing [21]. While 5G introduced Software-Defined Networking (SDN), Network Function Virtualization (NFV), and network slicing to support diverse vertical services, 6G will build on these by adding dynamic capabilities to NFV and network slicing. Comprehensive details on 6G enabling technologies can be found in references [3, 30, 21, 35].

In conclusion, 6G networks represent the next evolutionary step in wireless communication, aiming to meet the increasingly complex demands of a hyper-connected world. Unlike its predecessor, 5G, which introduced significant advancements such as mm-Wave technology and the Internet of Everything (IoE), 6G will push the boundaries further by integrating terrestrial and non-terrestrial networks to achieve seamless global coverage. The envisioned capabilities of 6G include ultra-high data rates, with peak speeds reaching up to 10 Tbps, and extremely low latency to support mission-critical applications. Additionally, 6G will rely on advanced technologies like the Terahertz (THz) spectrum, ultra-massive MIMO, and AI-driven optimization to handle the unprecedented density of connected devices and ensure pervasive connectivity.

6G will also introduce new usage scenarios, enhancing existing ones to cater for emerging applications that require high reliability, low latency, and massive data throughput. The integration of AI and ML into the network architecture will enable a truly intelligent and adaptive system, capable of supporting diverse vertical industries and ensuring secure, efficient communication through innovations like blockchain and quantum computing. Ultimately, 6G aims to not only extend the capabilities of 5G but also to unify and enhance them, creating a versatile, high-performance network that is ready to meet the challenges of the 2030-and-beyond digital landscape.

2.4 Related Works:

As the telecommunications landscape continues to evolve, an increasing number of services with stringent quality-of-service (QoS) requirements are emerging, placing significant strain on scarce radio resources. Effective bandwidth allocation has become a critical challenge in addressing these demands. This section reviews previous works on bandwidth allocation schemes, with a particular focus on both predictive and non-predictive approaches.

2.4.1 Non-Predictive Bandwidth Allocation Schemes

Non-predictive bandwidth allocation schemes are schemes that do not incorporate any form of forecasting of future resource requirements. These schemes use real-time information such as active user count, predefined thresholds to make admission decisions.The simplest bandwidth allocation scheme that incorporates no prediction mechanism is *complete sharing*. In this scheme, all traffic has unrestricted access to network's bandwidth [37].It has efficient utilization of resources as all resources can possibly be used by all incoming traffic. However, unrestricted access to network resources results in high call-blocking probability. Since dropping an ongoing call is generally more disruptive than blocking a new call [38], much attention in literature has been focused on prioritizing hand-offs over new calls.

Non-predictive bandwidth allocation schemes that prioritise handoff calls are *complete partition* and some *fixed or dynamic threshold or guard-channel* bandwidth schemes. In *complete partitioning*, network bandwidth is divided into separate units for different traffic, while for threshold or guard-channel schemes, a threshold value is set for which new calls are blocked when used bandwidth exceeds that threshold. The threshold value can be dynamic or static. For static threshold value, this often leads to underutilization of resources [39] and high new call blocking probability [40]. Consequently, considerable work on non-predictive threshold-based bandwidth allocations has been on dynamic or adaptive guard channel schemes.

There are number of dynamic or adaptive guard-channel schemes proposed in literature that are non-predictive in nature. In [39], Kulshrestha et al. propose an adaptive fractional guard channel algorithm for heterogeneous traffic, admitting new calls probabilistically. When the threshold values has been exceeded, new call access channels initially reserved for handoff calls with some probability. While this reduces blocking probabilities compared to fixed threshold scheme, its non-predictive nature makes determining thresholds difficult. The model assumes all traffic requires equal channels, which oversimplifies networks with diverse services. Additionally, dividing channels equally among traffic classes can waste resources, especially if certain traffic has lower arrival rates, making it unsuitable for more dynamic systems like 5G.

In [40], Mandour et al. propose a dynamic channel allocation scheme that prioritizes handoff calls by adjusting reserved channels,C_r, using a sensing parameter α based on traffic, unlike Kulshrestha et al.'s probabilistic approach. They compared their proposed algorithm to complete sharing and their dynamic scheme outperforms complete sharing in reducing handoff call dropping probability, highlighting the drawbacks of unrestricted access to resources. However, like [39], it suffers from underutilization of resources since C_r cannot be optimally set without prior knowledge of incoming traffic. This non-predictive nature limits its application in highly dynamic environments like 6G

Non-predictive schemes proposed in [39, 40] allocate available channels between handoff and new calls, prioritizing handoff calls over new ones. However, in [41], Sanon and Joshi present a non-predictive admission scheme that categorizes traffic into real-time (voice and video) and non-real-time (data services), prioritizing real-time traffic over non-real-time. Unlike [39, 40], their approach reserves resources not only for handoff calls but also specifically for real-time traffic. Additionally, their algorithm dynamically degrades non-real-time resources when resources for real-time traffic are insufficient. This strategy significantly reduces handoff dropping probability, while the added priority for real-time traffic ensures the network meets stringent QoS requirements. Prioritizing real-time traffic is also justifiable, as degrading non-real-time traffic has a lesser impact on the user experience compared to real-time service degradation. For example, slowing down email delivery to enable smoother calls greatly enhances QoS satisfaction

Instead of blocking new calls once the threshold is reached, as done in [39, 40, 41], Candan [42] proposes a dynamic guard channel scheme that queues new calls when the total used bandwidth exceeds the dynamically set threshold, holding them until a free channel becomes available. This approach has the advantage of reducing call blocking probability. However, the delays experienced by calls in the queue make this method unsuitable for delay-sensitive services like 5G uRLLC. In [43], Alioua et al. present a non-predictive scheme that improves on the methods proposed by [39, 40, 41, 42] by incorporating a retrial policy for both new and handoff calls. When a new or handoff call arrives and no resources are available, their algorithm employs a channel 'browning' concept, allowing the call to retry its request.

In summary, non-predictive algorithms fail to optimize resource utilization effectively without compromising QoS requirements, as they lack the ability to anticipate incoming traffic. Some works in the literature has been dedicated to developing predictive bandwidth allocation schemes, mitigating these challenges. These are review in the next section.

2.4.2 Predictive Bandwidth Allocation Schemes

Predictive call admission control and bandwidth allocation schemes utilize predictive techniques such as time-series analysis, regression models, and neural networks. These schemes are generally more efficient than non-predictive approaches, though they come with higher computational costs and potential for error [44]. Their efficiency is highly dependent on the accuracy of the predictions—larger predictive errors can diminish performance. Common predictive aspects explored in the literature include user mobility patterns, network traffic (load), and resource demands.

Mobility-Prediction-based Bandwidth Allocation Schemes

Several mobility-based bandwidth allocation schemes have been explored in the literature. In [45], Fazio et al. employ the Mobility Reservation Protocol (MSRVP) to predict user mobility and make resource reservations in advance. In this approach, each cell sends a PASSIVE_RESERV MSRVP packet to signal adjacent cells about incoming traffic. This method is effective because it enables adjacent cells to accurately reserve bandwidth for the expected network load. However, the signaling overhead can degrade overall network performance, as it requires additional channels to transmit control signals—channels that could otherwise be used to serve network traffic.

Scheme in [45] predicts only where the next call will be likely handed off neglecting the time of handoff. This oversight could lead to false signaling, causing adjacent cells to reserve resources prematurely, before the hand-off actually occurs. In [46], Yu and Leung propose a mobility-prediction-based bandwidth allocation scheme inspired by data compression techniques that not only predicts future location of mobile users, but the exact time when that will occur. They argue that a good data compressor inherently includes a predictive element to achieve effective compression.

In data compression, a dataset is broken down into a sequence of events and encoded using as few bits as possible. If a data compressor predicts the next character with high probability, it assigns that character a relatively shorter code. Therefore, if the overall code length is short (indicating good compression), it must also have been a good predictor. Their mobility prediction algorithm draws from Ziv-Lempel data compression algorithms, where user events $N, H_1, H_2, ...H_n$, representing a sequence of cells the user is likely to traverse, are treated similarly to substrings in Ziv-Lempel compression. Additionally, the scheme includes a mechanism to predict when an actual call will be handed off, making it more robust to false signaling.The draw back of this scheme is high computation burden.

While [46] utilizes learning theory derived from data compression, the schemes in [47, 48] leverage machine learning techniques. In [47], Kumar et al. use artificial neural networks to accurately predict users' future locations, enabling timely resource reservations. Their scheme trains a neural network using data obtained from the Home Location Register (HLR) and Visitor Location Register (VLR). The results show that the network can achieve 100% bandwidth utilization in some instances.

In [48] , Belhadj et al. employ a Long Short-Term Memory (LSTM) deep neural network to make accurate next-cell predictions based on vehicle mobility in 5G machine communication Internet of Things (mc-IoT). Their algorithm leverages on user trajectories derived from historical mobility patterns. Additionally, they demonstrate that accurately predicting the most likely future cell of users can assist in efficient slice resource allocation, particularly for sensitive 5G slices such as mMTC. However, these machine-learning-based mobility prediction algorithms collectively suffer from high computational costs and long training times.

The mobility-prediction-based schemes reviewed above suffer from several common drawbacks. First, they all rely on historical mobility data to make accurate predictions. This means that unpredictable mobility patterns, such as those of new subscribers, can significantly reduce the effectiveness of these schemes. Second, they face scalability challenges. As the number of users and mobility events increases, these algorithms incur higher computational costs, which can impact performance. Lastly, predicting only user mobility may not be sufficient in 6G environments. Since 6G will integrate both terrestrial and non-terrestrial networks, mobility will involve not only be of users but also moving non-terrestrial elements, such as satellites and Unmanned Aerial Vehicle (UAV)-assisted networks.

Instead of predicting individual user mobility, some predictive schemes proposed in literature predict the overall incoming traffic to make resource reservations. These schemes are reviewed next.

Traffic Prediction-based Bandwidth Allocation Schemes

In addition to user mobility, traffic flow is another crucial factor influencing resource allocation in wireless cellular systems. Efficient traffic management can significantly enhance network resource utilization [49, 50, 51]. Consequently, traffic prediction has gained considerable attention in the research community as a means to facilitate effective resource allocation. Various methods have been proposed for network traffic prediction, with the most popular techniques employing statistical algorithms, such as Auto-Regressive Integrated Moving Average (ARIMA), or machine learning algorithms like Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs).

In [49], Xiao and Chen use LSTM model to predict traffic and make optimized resource allocation based on there predictions. They model resource allocation as Knapsack problem and propose a greedy algorithm that allocates resources between eMBB,uRLLC and mMTC slices factoring in anticipated traffic. In a standard Knapsack problem, given the set of items each with weight and value, the goal is to find the number each item to include in collection such that the total weight is less than or equal to maximum predefined value. In their model, values and weight represent priority and predicted traffic for each services, respectively. In additional, the Knapsack threshold is taken to be the total bandwidth of the network. The simulation showed that the algorithm generally achieved better resource utilization. However their, approach require large volumes of data to train the model, which is often difficult to acquire.

An improved traffic prediction model that requires less training data is proposed by Perifanis et al.[50]. They propose Federated Learning(FL) algorithm for 5G base station traffic forecasting. In FL, several small models are trained locally and uploaded into a central server where a large model is trained using these small model. This promotes extremely high privacy levels [50], which is an increasing concern to modern artificially-driven world.

While [52, 50] propose individual traffic prediction models, a comparative study by Azari et al. [51] introduced a traffic prediction framework that utilizes statistical, rule-based, and machine learning tools for proactive traffic prediction-based resource allocation. The statistical, rule-based, and machine learning algorithms used are Auto-Regressive Moving Average (ARIMA), Random Forests (RF), and LSTM, respectively. They investigated the performance impact of traffic on Root Mean Squared Error (RMSE). The results showed that as the standard deviation of the dataset increases, ARIMA outperforms LSTM. However, LSTM performs better than ARIMA for bursty traffic. They further extended their work and demonstrated that traffic forecasting can lead to improved resource management.

In their later work, Azari et al.[53] extended the framework proposed in [51] by adding Machine learning-powered Discontinuous reception(DRX) for energy saving. They further justify that in addition to improving bandwidth allocation based in traffic forecasting methods, introduction DRx parameters for online traffic prediction significantly reduce energy consumption.

A significant limitation of the schemes studied in [49, 50, 51, 53, 54] is that they are generally confined to terrestrial networks. In 6G, terrestrial and non-terrestrial networks will be integrated to provide ubiquitous coverage. A more effective traffic prediction scheme for 6G environments is proposed in [55]. The authors utilize mobile edge computing-enabled high-altitude platform stations (HAPS) to predict traffic from ground users to satellite networks. Additionally, they propose a dynamic scheduling strategy for resource control based on traffic demand prediction.

The traffic prediction schemes reviewed here rely heavily on machine learning approaches, which often require high computational resources and extended training time. Furthermore, predictive schemes studied thus far forecast factors that influence resource demands to make resource reservations. For instance, schemes in [45, 46, 47, 48] use mobility patterns, while those in [49, 50, 51, 53, 54] forecast traffic patterns to make resource reservations. Instead of predicting factors that influence resource demands, some schemes in the literature directly predict the network resource demands themselves. These schemes are reviewed next.

Resource-demand Prediction-based Bandwidth Allocation Schemes

While bandwidth allocation based on mobility and traffic prediction has significantly improved resource utilization, these approaches inherently suffer from two key problems. First, they rely on mobility or traffic as factors that impact resource demands and derive resource requirements from these. In real-time multimedia networks, a large number of factors can impact resource demands for future handoffs [38]. Second, most of these schemes follow a collaborative approach, where base stations send signal packets to neighboring base stations, signaling them of incoming traffic. This results in signaling overhead.

Instead of modeling the factors that impact radio resource demands, the authors in [38, 56] model resource demand directly. In [38], Tao et al. propose localized resource prediction schemes that predict handoff resources for each service class directly. These schemes are localized in that each base station dynamically determines future handoff resource demands using location information, which reduces signaling overhead, as seen in schemes proposed in [45, 46]. The proposed schemes leverage the Wiener Process and Time Series Analysis using the Auto-Regressive Moving Average (ARMA) model to forecast the resources required for handoff directly.

Wiener Process is a Markov process where only present values are relevant in predicting future values. However, future values may also be correlated with not only present but also past values. Tao et al. [38] use ARMA, a time series forecasting model, to account for this correlation. Compared to the Wiener Process, the authors showed that ARMA achieves higher accuracy. This is expected, as ARMA incorporates past values, providing additional information for more accurate future demand predictions.

In [56], Dias et al. also propose localized predictive bandwidth schemes that uses time series analysis to forecast future handoff call resource demands. While [38] leverages ARMA, Dias et al use ARIMA which is different from ARMA in that it is a differenced ARMA. Difference in in statistics is a technique used to transform dataset making in stationary[57]. In addition to ARMA, the authors also proposed Trigg and Leach(Exponential Smoothing) time series model. It has been shown that despite its simplicity, the Trigg and Leach method achieves performance similar to ARIMA.

Although the resource-predictive schemes proposed in [38, 56] demonstrate potential for enhanced network performance by localizing resource prediction to each base station, their reliance on time series analysis inherently limits their prediction accuracy. More recent approaches employ machine learning algorithms, which provide scalable and highly accurate resource prediction capabilities. In [58], Dubba et al. propose several regression models to predict resource requirements for network virtual functions. These models include ensemble machine learning techniques such as Adaboost, bagging, and ExtraTree, as well as traditional models like Lasso, Bayesian, and Poisson regressors. The key difference between ensemble methods and traditional machine learning is that traditional models rely on a single algorithm, while ensemble methods combine multiple predictive models to achieve improved accuracy.

As expected, the results show that ensemble models significantly outperform traditional regression methods. Notably, the Adaboost regressor achieved the lowest RMSE value. This is justifiable, as Adaboost is specifically designed to improve the accuracy and robustness of predictive models by iteratively incorporating models that correct errors from previous iterations. Furthermore, these highly accurate models are more resilient and adaptable to dynamic environments compared to time series analysis models.

Linear regression models proposed in [58] are often assumes linearity is data. In [59], Binghui et al. propose a predictive network slicing algorithm that predicts future bandwidth requirements using Unit Time LSTM(UT-LSTM). The primary difference between UT-LSTM and traditional LSTM is that UT-LSTM typically focuses on unit time-step dependencies while traditional LSTM is focuses at capturing short and long-term dependencies in data. In some applications, the objective of UT-LSTM is to emphasize capturing dependencies that happen within a specific, shorter temporal window or even a single time step, reducing the memory requirement. The proposed algorithm is compared to NeuralProphet, Transformer and ConvLSTM,and it is evident from the results that UT-LSTM consistently outperforms them.

Other Predictive Bandwidth Schemes

Predictive admission control and bandwidth allocation are not only limited to mobility, traffic, and future resource demand prediction. Other schemes exist that do not fall under these categories. For instance, Wu et al.[60] use classical ARIMA to forecast the number of potential handoff-dropping calls. This proposal improves on traditional methods, which often require significant signaling and rely on static or overly simplistic models. However, reliance on ARMA modelling may limit its effectiveness in scenarios with highly non-stationary traffic patterns,or in networks such as 6G with rapidly changing user behaviour. With the emergence of Large Language Models (LLMs), more recent works have attempted to use these in bandwidth allocation within wireless networks. Lee and Park [61] proposed an LLM-based resource allocation method that aims to maximize spectral efficiency. Additionally, they have shown that LLMs eliminate the need to build and train deep-learning-based models. They considered a simple resource allocation problem between two communication pairs and demonstrated that LLMs offer greater efficiency in resource allocation.

2.5 Conclusion of Literature Review

2.5.1 Summary of Key Findings

This literature review highlighted evolutionary path of wireless cellular networks from 1G to 5G which has been marked by continuous improvements in data rates, latency, capacity, and support for diverse applications. Each generation has built upon the strengths of its predecessors while addressing their limitations, driving innovation in mobile communications technology.

The review then provided the background for the next generation network. It has been illustrated that, although 5G network showed massive advancements in improving quality of services, it may not be able to meet the ever increasing stringent requirements of 2030 and beyond. As we look towards 6G and beyond, the goal remains to meet the ever-growing demands of our increasingly connected world.

Lastly, this literature review highlighted the ever evolving landscape of admission control and bandwidth allocation schemes. In particular, the review categorizes existing methods into predictive and non-predictive, underscoring the limitations of non-predictive approaches. Predictive schemes leveraging mobility, traffic and future resource demand prediction show more significant promise in enhancing network resource utilization and improving quality of service.

2.5.2 Research Gap in the Literature

While significant progress has been made in predictive call admission control and bandwidth allocation schemes, most existing research primarily focuses on mobility, traffic, and direct resource demand prediction. A critical research gap that remains is prediction of service duration. As 6G will integrate terrestrial and non-terrestrial networks,there is increased possibility of hand-off calls as result of high speed satellite networks. To address this gap, this project aims to develop a service duration prediction model that not only predicts call duration but also incorporates service duration into the call admission decision-making. By integrating service duration prediction, the project seeks to reduce the frequency of hand-offs in integrated terrestrial and non-terrestrial networks

Chapter 3

System Model

3.1 Introduction

With literature reviewed in the previous chapter, this chapter presents the system model of the proposed predictive admission control and bandwidth allocation scheme. First, the network model that integrates both terrestrial and non-terrestrial networks is discussed in Section 3.2. Next, Section 3.3 presents the details of the service duration prediction algorithm. In Section 3.4, a detailed description of how predicted duration is incorporated into call admission control is presented. Finally, this chapter concludes by description of performance metrics used to evaluate service duration prediction and call admission control algorithm in Section 3.5.

3.2 Network Model

The network system considered in this work is illustrated in Figure 3.1. It is a heterogeneous wireless network integrating both terrestrial and non-terrestrial components. Deployed in an urban area, the network encompasses Low-Earth Orbit Satellite (LEO-S), 5G macro-cell base station, and 5G femto-cell base station, all of which have overlapping coverage areas.

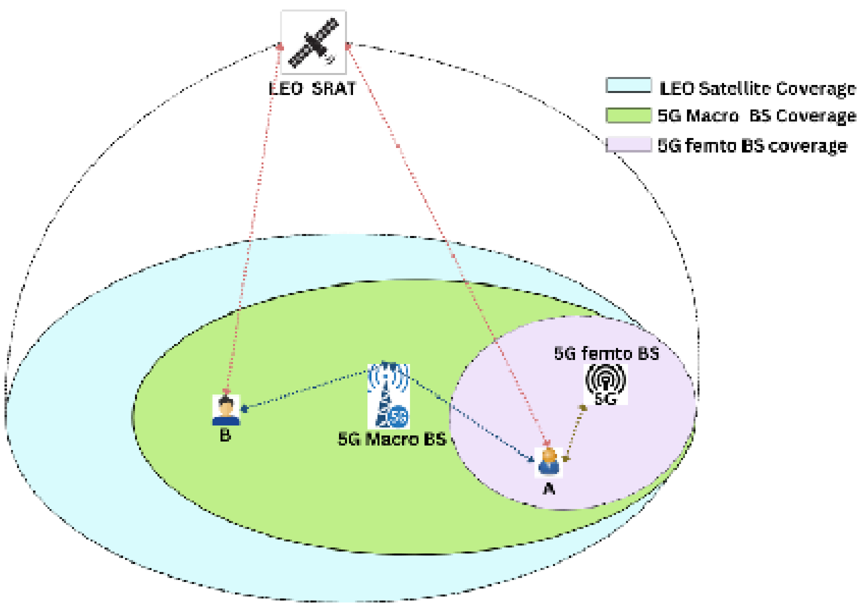


Figure 3.1: Integrated Terrestrial and Non-terrestrial Network Topology

The proposed predictive admission control and bandwidth allocation scheme consists of two main subsystems, as depicted in Figure 3.2. The *Duration-Predictor* subsystem manages the predictive aspect of the scheme. It takes, as input, a trained duration prediction model along with user data (i.e., input features for the model) and outputs the predicted service duration. This duration is then fed into the *Admission-Controller* subsystem, which also considers satellite visibility time, available RAT capacity, and the user's required basic bandwidth units. The output of this subsystem is the admission decision, which specifies the RAT where the user should be admitted.

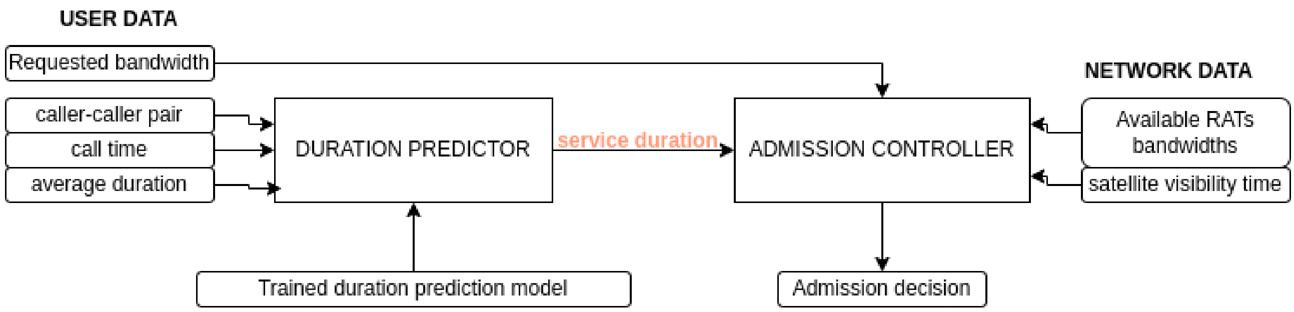


Figure 3.2: System block diagram of the proposed predictive admission control and bandwidth allocation scheme

3.2.1 Network Model Specifications

The network model is defined by the following specifications:

- The network comprises three RATs with overlapping coverage areas, each characterized by the following:
  1. **RAT-1:** A regenerative 5G-LEO satellite that hosts a fully functional gNB base station[62] operating at bandwidth of 10MHz with 15kHz sub-carrier spacing(SCS) [4].
  2. **RAT-2:** A Standalone 5G (SA-5G) macro base station with 25MHz of bandwidth and 15kHz sub-carrier spacing [63].
  3. **RAT-3:** a 5G femtocell base station with bandwidth of 20MHz and 15kHz sub-carrier spacing [64].
- There are two groups of users: Group A and Group B. Group A users can connect to all available RATs, while Group B users are out of femtocell coverage and can only connect to RAT-1 or RAT-2.
- The service provided by all RATs is voice calls.
- The radio resource in this work is resource block(RB), and the total capacity of RAT j, C_j , represents the total number of available resource blocks.

3.2.2 Network Model Assumptions

To simplify system implementation, and without loss of generality, the network model presented above makes the following assumptions:

- The service duration is assumed to be linearly dependent on caller-callee association, call time, and average duration.
- The call time is assumed to be normally distributed [65].

- Call arrivals follow a Poisson distribution [39, 66] with inter-call arrival time (time between individual calls) being exponentially distributed.
- The service duration follows normal distribution [67].
- The LEO satellite is assumed to have visibility time following beta distribution [68]. This distribution is justifiable because satellite visibility time is bounded by the minimum and maximum value at any point in time. Furthermore, beta distribution is flexible and one can obtain different shapes which could be tuned to obtain a more realistic distribution.

.

3.2.3 RAT Capacity Calculation

This subsection presents a procedure followed to calculate the capacity of each RAT using specifications from the previous subsection. Let $\lfloor . \rfloor$ be the floor operator, then the number of resource blocks in RAT j is given by $\left\lfloor \frac{B_j}{W_j} \right\rfloor$,where B_j and W_j is total bandwidth allocated and resource block bandwidth for RAT j [69], respectively. In the remainder of this paper, RAT-1, RAT-2, and RAT-3 will refer to the LEO-Satellite , 5G-macro BS, and 5G-femto BS radio access networks, respectively.

A 5G network resource block is made up 12 contiguous sub-carriers with spacing dependent on the numerology used [63]. Thus, all RATs have 12 contiguous sub-carriers per resource block. The capacities of RATs are therefore calculated as follows:

For RAT-1, the sub-carrier spacing is 15kHz, given 12 sub-carriers per RB we get C_1 as follows:

$$C_1 = \left\lfloor \frac{10 \text{ MHz}}{15 \text{ kHz} \times 12} \right\rfloor = 55$$

In RAT-2, the total bandwidth is 25MHz:

$$C_2 = \left\lfloor \frac{25 \text{ MHz}}{15\text{kHz} \times 12} \right\rfloor = 138$$

For RAT-3, allocated bandwidth is 20MHz:

$$C_3 = \left\lfloor \frac{20 \text{ MHz}}{15 \text{ kHz} \times 12} \right\rfloor = 111$$

The capacity values calculated above are maximum number of resource blocks for the given channel bandwidth. As [70] notes, not all of the channels are used for data transmission.For instance, some are used as guard channels to prevent co-channel interference. From Table 5.3.2-1 in [63], 3GPP defines the usable RBs for 10MHz, 25MHz and 20MHz FR1 channel bandwidths as 52, 133, and 109 respectively.

Thus, the actual capacities of RAT-1,RAT-2 and RAT-3 used in this work are $C_1 = 52$, $C_2 = 133$ and $C_3 = 109$, respectively.

3.2.4 Satellite Visibility Time Computation

With the assumption that LEO satellite RAT has altitude of 600 km [4], the visibility time of the satellite to any user is given by equations(3.1).

$$T_v = 2\beta\sqrt{\frac{(R_e + h)^3}{\mu}} \text{ [71]} \tag{3.1}$$

where R_e and h is radius of the earth and orbital altitude, respectively and $\mu = GM_e = 3.99054 \times 10^{14}$ is a constant. Assuming no restriction on angle of elevation, the satellite visibility angle β is defined by:

$$\beta = \arccos(\frac{R_e}{R_e + h}) \text{ [71]} \tag{3.2}$$

Given that $R_e = 6.37 \times 10^6$ meters, the calculated visibility time is found to be $T_v = 770\ seconds$. This work thus assumes maximum visibility time of the satellite to any use is 770 seconds. As such, this value is used to scale the output of random distribution where visibilities times are sampled from,since beta distributed outcomes are numbers between 0 and 1.

3.3 Service Time Prediction Model

Building on the network model described earlier, this section addresses how to predict the duration of phone calls. First, we define the problem and then explain the solution using a Generalized Linear Model (GLM).

3.3.1 Problem Formulation

The goal of this work is to predict the duration of a phone call based on various factors. We have a set of M Call Detail Records (CDRs), where each record contains the following information about the call:

- **Caller and Callee**: The unique identifiers(IDs) of who made the call and who received it.
- **Time of the Call**: Time of the day, when the call was made.
- **Average Duration**: The average duration of all calls made previously between these users.
- **Duration**: How long, in seconds, this call lasted.

To predict the duration of future calls, we use a Generalized Linear Model (GLM) [67], a statistical tool that can learn patterns in the data and map these patterns to predictions about the service duration. The GLM helps us establish a relationship between the input variables (such as caller, callee, call time, and historical average duration) and the response variable (call duration).

Formally, the prediction problem can be defined as follows:

- **Inputs**:
    1. Caller and callee IDs (p, q)
    2. Call time t

3. Average call duration between p and q, denoted as a.

- **Response/Output**: Service duration d, in seconds

The task is to build a model that takes these inputs and predicts the service duration d. The model will help the network factor in service duration in admission decision making.

3.3.2 Background to Generalized Linear Models

Generalized Linear Models are statistical models used to describe the relationship between the dependent variable and one or more independent variables when the dependent variable does not necessarily follow a normal distribution [72]. It differs from classical linear regression models in that the response variable can follow any of the family of exponential distributions [73]. Core to GLMs are three main components:

- **Error Distribution** - This component specifies the probability distribution of the response variable, [72, 73]. The choice of error distribution is completely dependent on the nature of the response variable and is critical to the model's coefficient approximation and accuracy.

- **Linear Predictor** - This component of GLM describes the linear relationship between the response variable and independent variables, also known as predictor variables. It is given by equation (3.3).

$$\eta = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p \text{ [72]} \tag{3.3}$$

  where η is the linear predictor,β_0 is the intercept term, β_i are coefficients(weights), and x_i are corresponding predictor variables.

- **Link Function:** This is the most important component of the GLM that connects the predictor η to the mean μ =E[Y] of the response variable. It transforms the expected value of the response variable to the scale of the linear predictor and ensures that the model's prediction stays within the range appropriate for the response variable's distribution. The GLM provides that the link function is given by equation (3.4).

$$g(\mu) = \eta \text{ [72]} \tag{3.4}$$

3.3.3 Service Prediction using GLM

This work derives the idea of using generalized linear model to predict service duration from [67]. In our case, the input variables are caller-callee pair, (p, q), the call time t, and average duration a. The response variable is service duration d, in seconds. From the problem statement, we note that our dataset contains M call detail records(or number of observations). Now, let an $M \times 1$ duration vector be $D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{bmatrix}$. From the dataset also, we can form an $M \times 4$ matrix of input features, $X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ \vdots & \vdots & \vdots & \vdots \\ x_{M1} & x_{M2} & x_{M3} & x_{M4} \end{bmatrix}$, where x_{j1}=p is the caller's ID,

$x_{j2} = q$ is the callee's ID,$x_{j3} = t$ is the time of the day and $x_{j4} = a$ is the average duration for previous calls made between users for the j^{th} call record.

Furthermore Let $\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_M \end{bmatrix}$ be an M vector of regression errors [74], and $\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$ be a vector of coefficients(or weights), then we can define the linear model as follows:

$$D = X\beta + \epsilon \text{ [75, 74]} \tag{3.5}$$

Since this works assumes the duration d is normally distributed,we use identity function $\mu = g(\mu)$ as our link function in equation (3.4) [67, 72] to obtain:

$$\mu = \mathrm{E[D]} = X\beta \text{ [67]} \tag{3.6}$$

, where D is given by equation (3.5) and E[.] is the expected-value operator.

3.3.4 Coefficients Approximation Using Ordinary Least Squares

The goal for the GLM is to learn the coefficients β. In this work, we use Ordinary Least Squares(OLS) algorithm to approximate these coefficients. As [76] notes, OLS estimates the coefficients β by minimising the sum of the squared residuals. Residuals are differences between the observed and model's predicted values of the response variable. This was found to be the most suitable algorithm because the model is to be trained offline and used during simulation to predict service duration of the incoming call.

Let $e = D - X\beta$, be the vector of residuals (note $e \neq \epsilon$). we can define the sum of the squares of this error vector as:

$$\begin{aligned} e'e &= (D - X\beta)'(D - X\beta) \\ &= D'D - 2D'X\beta + \beta'X'X\beta \text{ [74, 75]} \end{aligned} \tag{3.7}$$

, where $'$ is the transpose operation.

To minimize the sum of the squared residuals, $e'e$, OLS differentiates $e'e$ with respect to the estimator β and sets the result to zero:

$$\left.\frac{\partial e'e}{\partial \beta}\right|_{\hat{\beta}} = -2X'D + 2X'X\hat{\beta} = 0 \text{ [74]} \tag{3.8}$$

This results in the 'normal equation' $X'D = X'X\hat{\beta}$ [74]. The the approximated coefficients, $\hat{\beta}$ that minimize the squared error are given by:

$$\hat{\beta} = (X'X)^{-1}X'D \tag{3.9}$$

The next section,describes how this service duration is incorporated in admission control decisions.

3.4 Call Admission Control

Using the predicted service duration from the model described in previous section, this section now focuses on the call admission control algorithm, which is responsible for efficiently allocating network resources, factoring in the service duration in decision making.

The proposed call admission control algorithm admits users based on service duration such that frequency of satellite hand-offs are reduced. The predictor algorithm presented in the previous section is trained offline and during network simulation, it is used to predict duration of each user as they arrive and the output is fed into admission controller(see Figure 3.2) which makes admission decision.

The flowchart in Figure 3.3 shows the admission control algorithm proposed. When a new call arrives, the Admission Controller uses the duration predictor to predict service duration and obtains satellite visibility time. If the duration is less than satellite visibility time, the goal is to admit the user in RAT-1 since it will not experience satellite handoff. However, if duration is greater than satellite visibility time, the user would experience satellite handoff. In this case, the admission controller prioritizes admitting the user in terrestrial networks(RAT-2 and RAT-3 for group A users and RAT-2 for group B users).

For the case when the duration is greater than satellite visibility time, and terrestrial networks can not accommodate incoming call request, the call will be admitted in RAT-1, despite guaranteed possibility of satellite handoff. The call is blocked if there are not enough resources in all RATs that particular user can connect to.

For clarity, suppose an incoming call is predicted to last for 180 seconds, while the satellite is estimated to be visible to the user for 200 seconds. In this case, the admission controller will prioritise admitting this user into RAT-1. However, if predicted duration was greater than visibility time, say 240 seconds, the priority will be to admit the user in terrestrial networks, because the user would experience satellite handoff.

3.5 Performance Metrics

3.5.1 Prediction Model Performance Evaluation

There are a number of metrics that can be used to evaluate regression models, and there is no ‘one size fits all’ approach when it comes to choosing the right metric. The choice is entirely dependent on the problem being considered and the context. However, it is often beneficial to use a combination of metrics to get a more comprehensive view of the model’s performance. In this paper, the performance metrics used to evaluate the generalised linear model described in section 3.3.2 are Mean Absolute Error(MAE), Mean Square Error(MSE), and the R^2-score explained in [77]. While small values of mean absolute error and mean square error indicate better model performance, larger values of the R^2 score designate better model fit and generalization.

Mean Absolute Error (MAE)

This measures the average difference between the actual and predicted values. It has the advantage of being robust to outliers and outputting results in the same units as the response variable. Its formula is given by:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \text{ [77]} \tag{3.10}$$

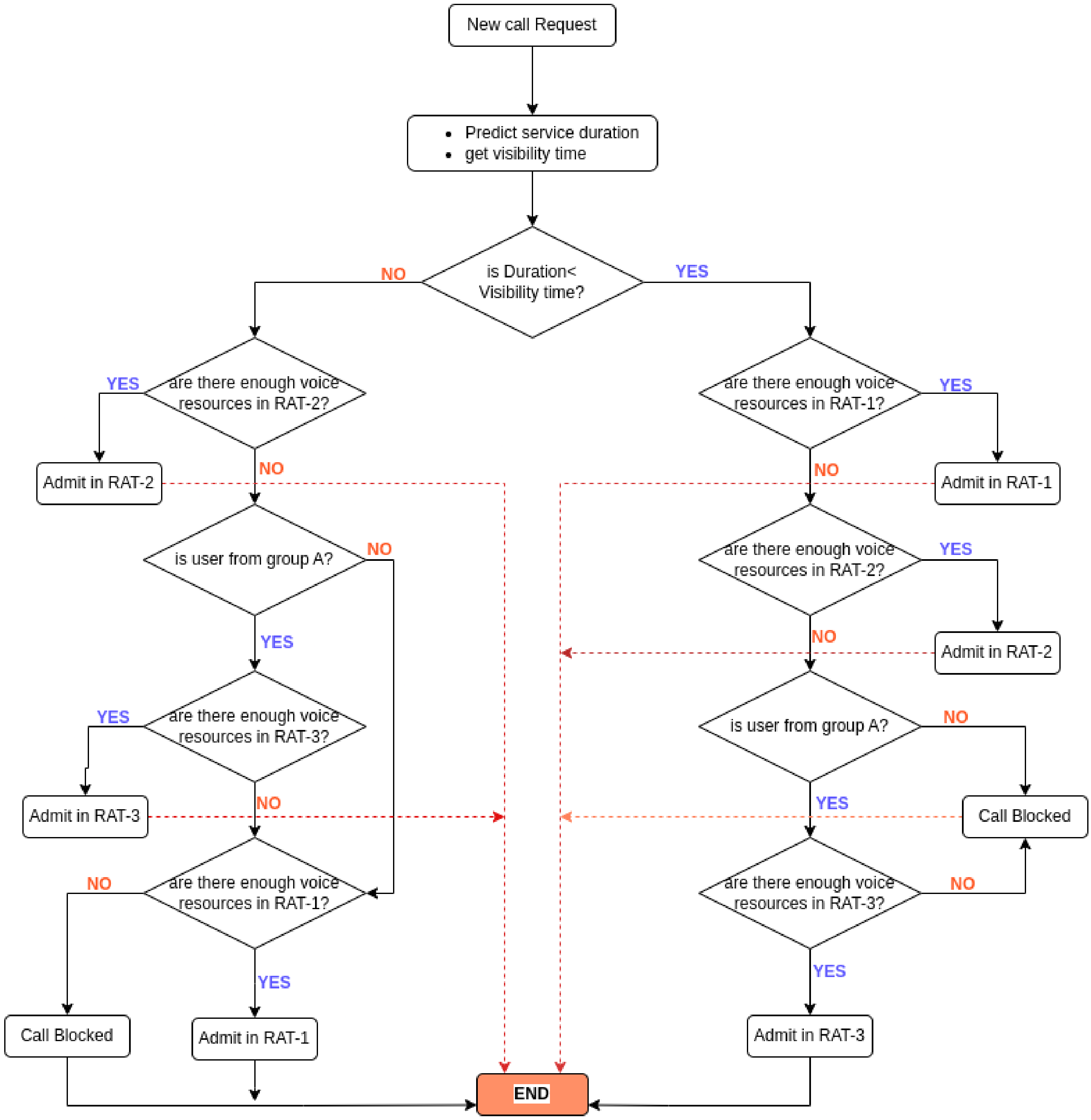


Figure 3.3: Flowchart for admission control algorithm

Mean Square Error (MSE)

The MSE measures the mean of the squared differences between the actual and predicted values. It is by far the most widely used performance metric in evaluating regression models. However, it has the disadvantage of penalizing outliers much more than the MAE and not producing results in the same units as the response variable.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \; [77] \tag{3.11}$$

R-Squared Score

Unlike MAE and MSE, the R^2 metric is a unit-less performance indicator that is independent of the scale of the data. It does not assess the model's performance in terms of loss, as MAE and MSE do. It is sometimes called the 'goodness of fit' because it measures how well the model fits the dataset.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y}_i)^2} \text{ [77]} \tag{3.12}$$

3.5.2 Network Performance Evaluation

The admission control algorithm implemented in this paper admits users into the network based on resource availability and predicted service duration relative to satellite visibility time. One of the primary objectives is to reduce the frequency of satellite hand-offs. The metrics used to evaluate the performance of the proposed algorithm are call blocking probability(CBP) and satellite handoff probability(SHP). The equations to find CBP and SHP are developed based on those used by Senouci el al. [78] to calculate handoff dropping probabilities for their algorithm.

New Call Blocking Probability

Depending on the availability of resources, an incoming call request may be blocked or admitted into the network. The call blocking probability (CBP) is the probability that a new call is blocked when a call attempt is made. The CBP is given by equation (3.13):

$$CBP = \frac{Number\ of\ calls\ blocked}{Total\ Number\ of\ call\ attempts} \tag{3.13}$$

Satellite Handoff Probability

To quantify how well the network reduces the frequency of satellite handoffs, the Satellite Handoff Probability (SHP) performance metric is used. This is the probability that a call admitted into the Satellite Radio Access Network will experience at least one satellite handoff. The SHP is given by equation (3.14).

$$SHP = \frac{Number\ of\ calls\ that\ experienced\ Satellite\ Handoff}{Total\ Number\ of\ calls\ admitted\ into\ Satellite\ RAT} \tag{3.14}$$

3.6 Conclusion

The chapter presented the proposed predictive call admission control and bandwidth allocation scheme's system model. This scheme has two subsystems: the *Duration Predictor* and the *Admission Controller*. The *Duration Predictor* uses a Generalized Linear Model (GLM) to estimate service duration. The estimated durations are fed into the *Admission Controller*, which makes the decision to admit the call or not. We will use MAE,MSE, and R^2 to evaluate the predictive model and CBP and SHP to evaluate the admission control scheme proposed. The next chapter provides detailed description of how this system was implemented in software,and how simulation results were generated.

Chapter 4

System Implementation

4.1 Introduction

This chapter builds upon the system model discussed in Chapter 3. Specifically, we provide detailed description of how the proposed system was implemented in software and how simulation results were generated. The software implementation of the proposed system is divided into three main components: *data generation*, *predictive model*, and *network simulation*. We begin by discussing the simulation software used, along with the rationale for its selection, in Section 4.2. Section 4.3 covers the synthetic call detail record generator, which is responsible for producing the data used in this study. Following that, Section 4.4 presents the implementation of the predictive model and outlines the data prepossessing techniques applied to the generated dataset. Finally, Section 4.5 describes how the network model was implemented and simulated.

4.2 Simulation Software

4.2.1 Rationale for Simulation Software

The proposed admission control technique was implemented in Jupyter Notebook using Python programming language. There are many good reasons for why Python turned out to be best fit for this project. To begin with, there is huge number of scientific libraries that are offered for use such as NumPy, Ski-learn, Pandas and SciPy, for data handling and prepossessing as well as implementing machine learning algorithms used in this work. Secondly, Python's simple syntax facilitates rapid testing and implementation of low-cost algorithms. This is very critical in carrying out research or in any other project where there is a lot of trial and errors using cheap algorithms, models and optimizations. Thirdly, Python arguably has one of the biggest programmer support available in the form of extensive resources, codes, tutorials, and forums for the optimal use of the language. Finally, it comes with powerful plotting libraries such as Matplotlib, Seaborn, and Plotly that can generate various types of plots for ease of visualization of results.

4.2.2 Simulation Environment

The Python version used to implement and test algorithm is `3.10.12` while simulation results were produced using Jyputer Notebook version `6.4.8`. These were installed and run on a computer with specifications shown in Table 4.1:

Table 4.1: Specifications of the computer used for system implementation

Parameter	Specification
Hardware Model	Dell Inc. Latitude 3310
Memory	8.0GiB
Processor	Intel Core i3-8145U CPU @ 2.10GHz×4.
Graphics	Mesa Intel UHD Graphics 620(WHL GT2)
Disk Capacity	256.1GB
OS Name	Ubuntu 22.04 LTS
OS Type	64-bit
GNOME Version	42.9
Windowing System	X11

4.3 Call Detail Record Generator

In this work, a synthetic Call Detail Record (CDR) generator was developed to produce data for training and evaluating the generalized linear model presented in Section 3.3. CDRs details information such as caller, callee call start time,call end time, caller's location e.t.c [79]. The implementation was adapted from the synthetic CDR generator by Songailaite and Krilavicius [65]. This section first provides an overview of their CDR data generator, followed by a detailed explanation of how a custom CDR generator was implemented based on their approach.

4.3.1 Overview of CDR Generator

In [65], Songailaite and Krilavicius developed a synthetic Call Detail Record (CDR) generator based on an analysis of a real telecommunication dataset. They examined the calling patterns of 20,000 customers over six months using statistical methods to extract key insights. Each CDR in the dataset included *date, caller ID, receiver ID, destination name*, and *call duration*. Their generator was designed to reproduce all of these features.

From their analysis, it was determined that call durations followed a Weibull distribution [80] with a shape parameter of 0.61 and a scale parameter of 413.62, while call times followed a normal distribution with a mean of 15.74 and a standard deviation of 4.62. These statistical findings were incorporated into their generator to produce synthetic CDR data that closely resembled real-world records. More detailed information about their generator, along with a link to their repository, can be found in [65].

Although their CDR generator effectively generates datasets for realistic network scenarios, certain aspects needed to be adjusted to align with the requirements of this project. For example, the call duration distribution was changed from Weibull to a normal distribution to be compatible with the Generalized Linear Model (GLM) algorithm. This modification was necessary because the Weibull distribution is not part of the exponential family of distributions and, as noted in [73], GLMs can only predict response variables whose distribution belongs to this family.

Additionally, their generator accounted for multiple network operators, assigning users based on market share thresholds. In this work, however, a single network is considered where all users make calls within the same network, so this feature was removed. Features that were retained include setting the call time mean to 15.74 and the standard deviation to 4.62.

4.3.2 Custom Call Detail Record Generator

The call detail record dataset in this paper, adopted from [65], was implemented using two classes: `User` and `DataGenerator`. The implementation is based on the approach in [65], with modifications made to fit the specific needs of this project. This section first describes the `User` class, followed by an explanation of the `DataGenerator` class.

The User Class

The `User` class describes the behavior of the user in the network. The network consists of 100 users, each with a unique `ID={1,2,...,100}`. No two users can have the same ID. The new `User` class has the following constructor:

```
def  __init__(self,user_id,call_count=0):
    """
    Creates a new User object with the given ID, and initializes all parameters of the this User
    """
    self.ID =user_id

    #lits of contacts: users that this User can call, or be called by them
    self.contacts = []

    #distionary that stores relationship values of this User with its contacts
    self.relationships = {}

    #flag thta indicates that the user is in a call
    self.isBusy =False

    #call logs of all users this user called
    self.call_logs = {}
```

The `self.ID` is the unique identifier of the user, and `self.contacts` is the list of contacts (`User` objects) that this user can call or be called by. In this network, caller-callee pairs are assigned random values that determine their calling behavior. This is referred to as `relationship`. Caller-callee pairs with large `relationship` values are more closely related and make calls more frequently than pairs with smaller values. This `relationship` value takes a random value from `0-30`.

As a result, each `User` also maintains a dictionary of the relationships, `self.relationships`, that the user has with their contacts. The keys are the contact users' ID numbers, and the values are the relationship values. For clarity, suppose a user with ID `10` has three contacts with IDs: `[5,6,12]`, and corresponding relationship values of `[19,7,3]`. Then the `relationships` variable of user 10 will be: `relationships={5:19,6:7,12:3}`. From this relationship dictionary, we can see that user 10 is more closely related to user 5 (with a relationship value of 19) than to user 6 (with a relationship value of 7). This means that user 10 will make more frequent (and likely longer) calls to user 5 than to the other contacts. The `isBusy` flag is set to `True` when the user is in call. This is so that the user can not engage in multiple call simultaneously. This flag is set to `False` when the user's session ends.

Furthermore, each `User` maintains additional variable: `User.call_logs`. `call_logs` is the dictionary of all calls that the user has made with their contacts. It is keyed by the contact's user ID, and the corresponding value is a list of all call logs between the users. Each call log is an array that describes the properties of that call. Using example scenario above, suppose user-10 has made some calls with thier contact, the `call_logs` dictionary might look like:

```
call_logs={5:[call_log_1,call_log_2,...call_log_n],
6:[call_log_1,call_log_2,...,call_log_k],
12:[call_log_1,call_log_2,..,call_log_i]}
```

In this dictionary, `call_log_i`,is the list of call details between user-10 and the corresponding contact.

A user may add new contacts to their list of contacts using the following method:

```python
def add_contact(self,contact, relationship):
    """
    Adds the given contact into the calling user's contact list and returns True if the addition was
        successful. Returns False otherwise(e.g if contact already exists)
        """
    if self.isContact(contact) or contact.ID == self.ID:
        return False
    else:
        self.contacts.append(contact)
        self.relationships[contact.ID] = relationship
        return True
```

As can be seen, we first check to see if the contact has already been added, using the method `isContact`, defined as follows:

```python
def isContact(self,user):
    """
    Retuns True if the given user is in this User's contacts list.
    """
    for contact in self.contacts:
        if contact.ID  == user.ID:
            return True
    return False
```

If the contact already exists, we return `False` to indicate that the addition was not successful. Otherwise, we add the user to the list of contacts and return `True`.

From time to time, the user may select a contact at random (with probability determined by their `relationship` value) from their `contacts` list and make a call using the `dial_call()`. This method computes the selection probability based on relationship values and then selects the contact at random with that probability to make a call.If the selected contact is busy, the call user tries another contact for given random number of trials, after which the call initiation is aborted. If the call is successful, the resulting call log is then recorded in the `call_logs` variable of that user.

The DataGenerator Class

Recall that synthetic data generation is implemented across two classes: `User` and `DataGenerator`. Having described the `User` class, we now explain the `DataGenerator` class. This class defines a data generator object that produces synthetic call detail records for a given period of time (in this work, call detail records were generated for the month of January 2024). Each call detail record is an array of the form:

`[caller-ID,receiver-ID,relationship, call-type,timestamp,adjusted-time(h),avg-duration(sec),duration(se`

The record's entries are as follows:

- `caller-ID`: The unique identifier of the user who initiated the call.
- `receiver-ID`: The unique identifier for the user who received the call.
- `relationship`: The relationship value between the caller and the callee. As explained before, this value determines the calling behavior between the two, in terms of how long their calls last and how frequently they call.
- `call-type`: The type of the call made between the caller and callee (it is always `voice`, since our network model only supports voice calls).
- `timestamp`: The timestamp of the call (including the date and start time).
- `adjusted-time(h)`: The time difference between the call's start time and the network's peak time. For instance, if the call was made at 1:00 PM, and since the network's peak time is around 3:30 PM [65], the `adjusted-time(h)` would be approximately 2.5 hours.
- `avg-duration(sec)`: The average duration of all previous calls made between the caller and the callee. It is zero for the first call and accumulates for subsequent calls. This value significantly influences the call's duration.
- `duration(sec)`: The duration of the current call in seconds.

Since we are generating data for a regression model, we explicitly linked the call duration linearly to its predictors: `relationship`, `adjusted-time(h)`, and `avg-duration(sec)`, assigning weights to these variables to produce reasonable values for the duration. The duration of the first call is computed by the method `get_one_time_duration`, defined as follows:

```
def get_one_time_duration(relationship,time):
    """
    returns the duration of the first call that this user makes with the contact of given
        relationship value.
    """

    duration = 30.75*relationship+7.75*User.adjust_time(time)

    return 0 if duration <0 else duration #eliminate negative duration
```

For subsequent calls, we use the method `get_duration`, defined as follows:

```
def get_duration(relationship,time,avg_dur,noise_level=0.001):
    """
    Computes the duration of the current call that this user is about to make.
    """
    duration = 0.5*relationship+0.3*User.adjust_time(time)+0.95*avg_dur
    if duration <0:
        return 0
    else:
        eturn np.random.normal(duration,duration*noise_level)
```

The weight values of each predictor were tuned iteratively to obtain more realistic call durations. The call detail record generator in [65] produces durations with a mean of 340 seconds, while our model generates durations with a mean around 420 seconds (dependent on the number of samples produced). This is a reasonable average duration that resembles real network data as analyzed in [65]. Additionally, some random noise is added to the resulting duration values because real-world datasets are usually noisy.

To generate the full call detail record dataset, the `DataGenerator` class is used. This class maintains a list of network users. It has a method called `build_network`, which creates a list of 100 `User` objects and stores them in the `users` array. For each user, a random number of calls they can make per day is allocated using a Weibull distribution, as in [65]:

`int(np.random.weibull(0.74) * 11.13).`

Next, the method assigns each user a random number of contacts between `min_contacts` and `max_contacts`. We arbitrarily set `min_contacts = 10` and `max_contacts = 50` to create a densely connected network of users. After assigning contacts, the `build_network` method then assigns random `relationship` values to all contacts of each user. Once the network of users is built, they can start making calls.

To simulate the network environment where users make calls to one another, we defined a method of `DataGenerator` class called `generate_call_logs`. This method simulates the network over the given period and time, and records all calls that users made to one another. The call logs are later extract by the method `extract_call_logs`and saved to an external file.Over the period of January 2024, we generated 11601 call detail records for the. The next section describes how the predictive model was implemented. The generated dataset in analysed in chapter 5.

4.4 Linear Prediction Model

The linear prediction model used to predict service duration in this work is implemented by the class `Predictor`. This class has the following constructor:

```
def __init__(self,generator):
    self.model = None
    self.scaler = None
    self.dataGenerator = generator
```

where `model` is the actual prediction model object, `scaler` is a `StandardScaler` object used to scale the features to be standard normally distributed, and `dataGenerator` is an instance of the `DataGenerator` class, which generates data for this `Predictor` object. This section provides a detailed explanation of how the model was implemented. We begin by outlining the procedure for selecting and preprocessing the dataset, followed by details of the model's training and evaluation implementation.

4.4.1 Data Preprocessing

The generated call detail record dataset has a total of ten columns of information: `caller-ID, receiver-ID, relationship, call-type, timestamp, adjusted-time, avg-duration`, and `duration`. Among these, the input features to the model are `relationship`, `adjusted-time`, and `avg-duration`, while the response variable is `duration (sec)`. A standard procedure was followed to identify and remove redundant samples from the dataset.

First, all samples with `duration (sec)` less than or equal to 3 seconds were removed [67]. Since the goal was to train the model to learn the calling patterns between users, it was necessary to remove all samples corresponding to the first call between users. These samples are identified by an `avg-duration (sec)` of zero. This filtering was achieved using the following code:

```
data_filtered = data[(data['duration(sec)']>3) & (data['avg-duration(sec)']!=0)]
```

After this filtering, 8249 samples remained in the dataset from original 11601 samples. Next, the input features were scaled to improve training performance and convergence. The `StandardScaler` from `sklearn.preprocessing` was used for this purpose. It scales features to be standard normally distributed (mean of 0 and variance of 1). The `StandardScaler` was preferred over the `MinMaxScaler` because all input variables of the model (`relationship`,`adjusted-time`,`avg-duration`) followed a normal distribution.

After preprocessing the input features, the dataset was split into a 7:3 ratio, where 70% of the data was used for training and the remaining 30% for validation. The next subsection provides details on the model training and evaluation.

4.4.2 Model Training and Evaluation

The regression model used is `OLS` (Ordinary Least Squares) from `statsmodels.api`. This was preferred over `sklearn.model.LinearRegression()` due to its easier presentation of training statistics and parameter tuning. The model coefficients were approximated using the OLS algorithm, as discussed in Chapter 3:

```
self.model = sm.OLS(y_train, X_train_const).fit()
```

After training the model,training results were extracted as follows:

```
conf_intervals = self.model.conf_int()
coefficients = self.model.params
standard_errors = self.model.bse
p_values = self.model.pvalues
```

These results are presented and analyzed in Chapter 5.

Next, the model was evaluated on the remaining 30% of the dataset. The evaluation metrics used were Mean Squared Error (MSE), Mean Absolute Error (MAE), and the R^2-Score. In addition, residual plots and other performance visualizations were produced to assess the model's performance. All these results are presented in Chapter 5.

The next section describes how the proposed predictive admission control was implemented and simulated.

4.5 Network Simulation

4.5.1 Implementation

The network simulation is implemented using a class called `Simulator` which is discrete-event simulation framework. The simulation can be configured to run for a specific duration. The `simulate` function (detailed in the appendix) takes the following parameters: `start_hour` (the hour of the day when the simulation starts), `total_time` (the total number of hours the simulation runs), `arrival_rate` (the rate of new call arrivals), and `groupA` (the probability of a user belonging to Group A). Inter-call arrival times are sampled from an exponential distribution using `time_interval = np.random.exponential(1/arrival_rate)`.

During the simulation, the function maintains a list of active calls, `active_calls`, where each entry is a tuple in the form `(caller, callee, RAT, call_end_time)`. At each simulation step, calls that have ended before current time are removed using:

```
for call in active_calls:
    if call[3] <= current_time:
        call[0].isBusy = False # Free the caller
        call[1].isBusy = False # Free the callee
        self.RATs[call[2]-1] += Simulator.voice_RB
        active_calls.remove(call)
```

As shown in the code snippet, we iterate over all active calls and, for each call where `call_end_time` is less than or equal to the `current_time`, the `isBusy` flag of both the caller and callee are set to `False`, indicating they are no longer engaged in a call. Next, the bandwidth occupied by the call is released and the call is removed the call from the `active_calls` list. `Simulator.voice_RB` represents the number of resource blocks (RBs) used by the call, `RAT={1,2,3}` refers to the RAT-ID where the call was admitted, and `self.RATs` is a list that maintains the remaining capacity of each RAT during the simulation. When a call leaves the RAT, available capacity of that RAT is increased by the number of RBs previously occupied by the call, returning the RAT to its pre-call state.

After handling call termination, a new caller is sampled at random from the list of users and a new call is initiated using the `dial_call` function(see appendix) of the `User` class. This function selects a contact of the caller at random and attempts to establish a call, provided the contact is not busy. If all contacts are busy, the function returns `None`. Otherwise, it returns the callee and the average call duration between the caller and callee. These inputs, along with the current call time, are fed into the predictive model to estimate the call duration. The visibility time is then sampled from a beta distribution, and the `admission_control` function is invoked to execute the proposed admission control algorithm. This function returns the RAT-ID where the call is admitted or 0 if admission is denied.

If the call is successfully admitted, its end time is calculated using the predicted duration and it is added to the list of active calls. This process repeats throughout the entire simulation period.

4.5.2 Performance Data Acquisition

To effectively evaluate the performance of the proposed admission control and bandwidth allocation scheme. The simulation of the network was carried under difference scenarios o. Separate functions we defined to alter with the simulation flow and gather performance evaluation data, such as call blocking probabilities.

Experiment Investigating Significance of Service Duration in Admission Control

To appreciate the importance of considering service duration in admission control, the network was simulated for two admission scenarios: 1) *admission that does not consider call duration is admission decision*. The call is admitted in any RAT with enough Bandwidth and 2) *the proposed admission control algorithm which considers service duration in admission control*. The function that simulates the network and gathers all data on call blocking and satellite handoff probability is `effect_of_duration_prediction` defined as follows:

```
def effect_of_duration_prediction(start=12,time=5.0/18.0,groupA=1.0):
    CBP = []
    SHP = []
    CBP_test = []
    SHP_test = []
    sim = Simulator()
    for rate in Simulator.rates:
        sim.simulate(start,time,arrival_rate=rate,test=['prediction'],groupA=groupA)
        CBP.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
        CBP_test.append(sum(sim.blocked_test)/sum(sim.attempts_test) if sum(sim.attempts_test)!=0
            else 0)
        SHP.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
        SHP_test.append(sum(sim.handoffs_test)/sum(sim.rat_1_calls_test) if sum(sim.rat_1_calls_test
            )!=0 else 0)
    return CBP,SHP,CBP_test,SHP_test
```

As can be seen, this function takes the following arguments:`start`(time to start the simulation), `time`(simulation time), and `groupA`(proportion of group A users in the network). This function the arrays `CBP` and `SHP` store call blocking and satellite handoff probabilities of admission scenario when duration is considered, respectively. Conversely, the arrays `CBP_test` and `SHP_test` collect call blocking and satellite handoffs probabilities ,respectively. When running the `simulate` function, the `test` is set to `prediction` which will make the `Simulator` to gather the required data for this particular experiment.

Experiment Investigating Impact of Duration

This work investigated the effect that service duration has on the performance of the proposed admission control and bandwidth allocation scheme. The function that simulates and gathers data for this particular experiment is `effect_of_duratio` defined as follows:

```
    def effect_of_duration(start=12,time=5.0/18.0,grp=1):
    data =  {}
    scenarios = ['short','medium','long']
    sim = Simulator()
    for scenario in scenarios:
```

```
        CBP = []
        SHP =  []

        for rate in Simulator.rates:
            sim.simulate(start,time,arrival_rate = rate,test = ['duration',scenario])
            CBP.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
            SHP.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)

        data[scenario] = (CBP,SHP)
    return data
```

the `grp=1` means the simulation consisted of only group A users. For each `scenario` the function iterates in all arrival rates, and collects data on call blocking probability and satellite handoff probability.

Experiment Investigating Impact of Visibility Time

We conducted an experiment to investigate the effect that visibility time has on service duration. For this experiment, the network was simulated under 'short' (less than 200 seconds), 'medium' (between 200 seconds and 400 seconds) and long (more than 400 seconds) visibility time scenarios. The function that gathers data for this specific experiment is `effect_of_visibility_time` defined as follows:

```
def effect_of_visibility_time(start=12,time=5.0/18.0,grp=1):
    data = {}
    scenarios = ['short','medium','long']
    sim = Simulator()
    for scenario in scenarios:
        cbp = []
        shp = []
        for rate in Simulator.rates:
        sim.simulate(start,time,arrival_rate=rate,test=['visibility',scenario],groupA=grp)
        cbp.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
        shp.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
        data[scenario] = (cbp,shp)
    return data
```

Experiment Investigating Impact of User Group

Since the network supports two different user groups, the experiment was conducted to investigate how each of these different groups impact the performance of proposed algorithms. The simulation was run under different duration scenarios and the data for each scenario was gathered by the function `effect_of_user_group` defined as follows:

```
def effect_of_user_group(start=12,time=5.0/18.0):

    groups = {'100%-A':1,'80%-A':0.8,'20%-A':0.2,'0%-A':0}
    data = {}
    sim = Simulator()
    for g in list(groups.keys()):
```

```
        cbp = []
        shp = []
        for rate in Simulator.rates:
        sim.simulate(start,time,arrival_rate=rate,groupA=groups[g])
        cbp.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
            shp.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
        data[g] = (cbp,shp)
    return data
```

4.6 Conclusions

In this chapter, we detailed the implementation of the system model used for the simulation. The complete code for the implementation and the generation of results can be found in the Appendices. The next chapter presents and analyzes the results of simulating the proposed admission control scheme under various scenarios.

Chapter 5

Results and Discussion

5.1 Simulation Parameters

The simulation parameter of this work are summarised in Table 5.1.

Table 5.1: Simulation Parameters

Parameter(s)	Description	Value(s)
N	number of users in the network	100
$\mu_{call\_time}, \sigma_{call\_time}$	mean and standard deviation of call time	15.74, 4.62 [65]
C_1, C_2, C_3	Total capacity,in RBs, of RAT-1,RAT-2 and RAT-3	52, 133 ,109
B_1	required RBs for voice calls	1bbu [81]
λ	new call arrival rates	0.5,1.0,1.5,...,5
T_{min}, T_{max}	minimum and maximum visibility time	0, 770 seconds

5.2 Data Analysis

The procedure of generating synthetic data for training and evaluating the predictive model was explained in Chapter 4, Section 4.3. Each record has *caller-ID,receiver-ID,relationship,call-type,call-time,adjusted-time,avg-duration(sec)* and *duration*. Each caller-ID and receiver-ID are unique identifiers for the callers and callees, respectively. The *relationship* is a value that measures the relationship between caller and callee and determines the frequency and duration of the calls made by the corresponding caller-callee pair. The *adjusted-time* is the time difference between the call-time and the network peak calling time of 15.74 [65]. The *avg-duration* is the cumulative average of the duration of all previous calls between the associated caller-callee pair.

The first 5 and last 5 samples of the generated dataset, sorted in ascending order of ’timestamp’, are shown in Figure 5.1.

Using `pandas.DataFrame`’s `describe` method,we extracted the statistics of the generated dataset, and the results are shown in Figure 5.2. In this figure,we can see along the row named `count` that a total of 11601 samples are generated,and there is no column with missing values.

The rows named `25%`, `50%`, and `75%` give percentiles of each attribute of the dataset. They describe the percentages of dataset in the column that are less than or equal to the given value. For instance, the `25`-percentile of duration is `328.4788` seconds. This means that 25% of all durations in the dataset are `328.4788` or less.

Important to this work on Figure 5.2 are the mean and standard deviation of `418.0431` and `132.3384` seconds, respectively, of service duration. These values will be used the divide call duration into categories as will be

caller-ID	receiver-ID	relationship	call-type	timestamp	adjusted-time(h)	avg-duration(sec)	duration(sec)
72	29	7	voice	2024-01-01 00:01:40.337381	15.71222	0.00000	337.01972
100	93	8	voice	2024-01-01 00:04:14.036422	15.66944	0.00000	367.43819
30	3	10	voice	2024-01-01 00:05:24.107572	15.65000	0.00000	428.78750
22	1	13	voice	2024-01-01 00:08:25.484437	15.59972	0.00000	520.64785
92	61	16	voice	2024-01-01 00:10:33.132921	15.56417	0.00000	612.62229
...	...	...	...	...	...	...	...
2	17	15	voice	2024-01-31 23:23:36.728228	7.65333	481.98708	465.15835
95	79	13	voice	2024-01-31 23:24:37.995078	7.67028	422.78517	409.91700
9	100	14	voice	2024-01-31 23:24:55.874732	7.67528	520.32147	501.43877
91	67	12	voice	2024-01-31 23:25:31.932791	7.68528	439.93278	426.03508
44	73	11	voice	2024-01-31 23:34:43.018438	7.83861	373.09410	364.30181

Figure 5.1: First 5 and last 5 samples of generated dataset, sorted in ascending order of timestamp

	caller-ID	receiver-ID	relationship	timestamp	adjusted-time(h)	avg-duration(sec)	duration(sec)
count	11601.000000	11601.000000	11601.000000	11601	11601.000000	11601.000000	11601.000000
mean	50.549694	47.710887	12.150504	2024-01-15 21:27:34.798359552	6.571092	315.379068	418.043107
min	1.000000	1.000000	0.000000	2024-01-01 00:01:40.337381	0.000280	0.000000	33.953330
25%	26.000000	22.000000	9.000000	2024-01-08 10:50:05.736015872	2.884720	0.000000	328.478820
50%	50.000000	47.000000	12.000000	2024-01-15 22:47:40.716834048	5.905830	370.877960	413.894030
75%	76.000000	73.000000	15.000000	2024-01-23 09:33:29.421617920	9.870000	486.952090	505.480310
max	100.000000	100.000000	26.000000	2024-01-30 23:54:51.354282	15.739440	877.260760	877.260760
std	28.981292	28.678741	4.521436	NaN	4.382349	229.447181	132.338260

Figure 5.2: Statistics of the generated dataset

seen in the subsequent sections.

The resulting histogram of duration is shown in Figure 5.3. From these histogram,it is evident that service duration is normally distributed(as was assumed), with most of the calls lasting between 300 and 600 seconds.

5.3 Model Training Results and Analysis

The dataset generated by the call detail record generator was used to train and evaluate the performance of the generalized linear model. From the data samples shown in Figure 5.1, the inputs to the model are *relationship*, *adjusted-time*, and *avg-duration (sec)*, while the response variable (output of the model) is *duration (sec)*. These inputs are also called predictors. Recall that service duration is linked to its predictors linearly by:

$$\text{duration} = \beta_1 \times \text{relationship} + \beta_2 \times \text{adjusted\_time} + \beta_3 \times \text{avg\_duration} + \epsilon,$$

where the ϵ term is essential to add randomness (noise), because real-life datasets are noisy.

Our goal has been to learn the coefficients β_1, β_2, and β_3. The data pre-processing procedure followed and the model implementation were discussed in Chapter 4, Section 4.4. The training statistics are summarized in Table

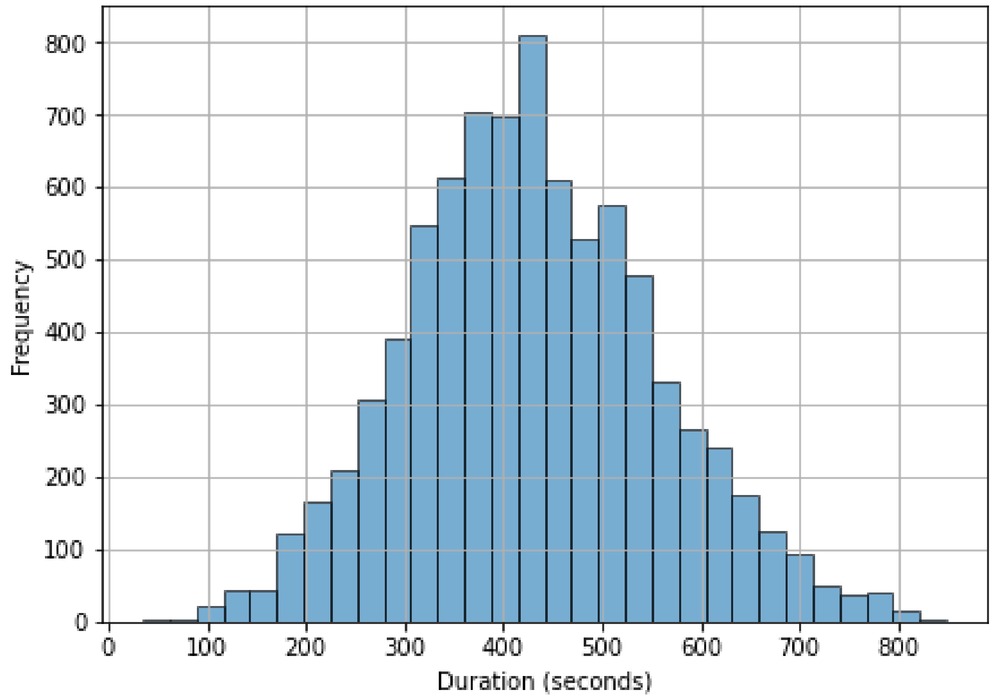


Figure 5.3: Histogram plot showing distribution of service duration

Table 5.2: Training results of Generalized Linear Model

Variable	Coefficient	SE	CI-lower	CI-upper	p-value
const	430.228635	0.029992	430.169840	430.287431	0.000000e+00
relationship	0.473028	0.054725	0.365748	0.580309	6.954348e-18
adjusted-time	1.339250	0.029993	1.280453	1.398047	0.000000e+00
avg-duration	125.107994	0.054725	125.000713	125.215275	0.000000e+00

5.2. In this table, the **coefficient** column provides the estimated values of $\beta_1, \beta_2, \beta_3$, as well as the constant term. The interpretation of these coefficients is that they represent the mean change in the response variable per unit change of the predictor, holding all other predictors constant. For example, the coefficient of the *relationship* predictor variable is 0.473028. This means, on average, the duration changes by 0.473028 seconds for each 1-unit change in the *relationship* value while other variables are held constant.

Surprisingly, the ordinary least squares algorithm included a constant term, although the original linear function did not have an intercept. The practical interpretation of this is that the response (duration) equals 430.228 seconds when all predictors are set to zero, which is clearly has no practical meaning. However, this constant is significant, and most regression algorithms include it because it leads to unbiased predictions [82]. OLS assumes that the response variable and the residuals have a zero mean. Most regression models include the constant term to ensure this assumption holds. Thus, the constant term serves to meet the model's assumptions rather than providing a practical interpretation.

Notably, the constant term and the *avg-duration* predictor are given the largest coefficients or weights. This means that the constant term and *avg-duration* contribute the most to the duration. To appreciate the significance of this, note that the coefficient of *avg-duration*, as visible from Table 5.2, is 125.107994. This means that, when all other predictors are held constant, a unit change in *avg-duration* causes a 125.107994-second change in the predicted duration.

The **SE** in Table 5.2 represents the standard error of the estimated coefficients. It quantifies how much the coefficient is likely to vary due to random sampling variation. A small standard error indicates that the coefficients are precise estimates, while large SE suggests more uncertainty about the coefficient's true value. As evident from Table 5.2, all coefficients have acceptably small standard errors. This means they are precise estimates, and we are more certain about their values.

The **CI-upper** and **CI-lower** indicate the upper and lower boundaries of the confidence interval. The confidence interval indicates the range within which the algorithm is 95% confident that the estimated coefficient lies within them. As can be seen from Table 5.2, our model obtained narrow confidence intervals. For instance, the 95% confidence interval for *avg-duration*, with a coefficient of 125.107994, is from 125.000713 to 125.215275—an interval of about 0.214. This suggests that the model is 95% confident that the coefficient of *avg-duration* lies within 125.107 ± 0.214. The short confidence interval also reiterates that our model is precise, as shown by the small **SE**s.

The last column of interest in Table 5.2 is the **p-value** column. This column provides the p-values of the estimated coefficients for each predictor variable. The p-value tests the null hypothesis that the coefficient is equal to zero (i.e., the predictor has no effect). Specifically, it estimates the effect that eliminating the predictor variable from the equation would have on the prediction results. For example, the null hypothesis for *relationship* would be: 'If the coefficient 0.473028 for *relationship* were zero, what effect would this have on the overall prediction?' A small p-value (less than 0.05 [67]) indicates that the predictor variable should not be excluded from the equation (i.e., rejects the null hypothesis).

From Table 5.2, we can see that the p-values of all coefficients are very small, with *const* and *avg-duration* having exact zeros. This suggests that all the variables are significant in determining the duration. This is expected because the response variable (duration) is explicitly defined as a linear combination of these predictors. However, in real telecommunication datasets, it might be possible that some predictor variables do not significantly affect the duration and thus would attain larger p-values.

5.4 Model Performance Evaluation Results and Analysis

The predictive model was trained on 70% of the dataset, and the remaining 30% was used for its evaluation. We used *Mean Squared Error (MSE)*, *Mean Absolute Error (MAE)*, and *R-Squared Score (R^2)* as evaluation metrics. The equations of these metrics are provided in Chapter 3, Section 3.5.1. The MSE measures the average of squared residuals, while MAE determines the average of the absolute values of residuals. R^2 is not a loss metric like MSE and MAE, and measures the variability in the response variable that can be explained by the predictor variables. In other words, it determines how well the model fits the dataset.

From the evaluation dataset, we obtained the MSE value of 4.965, MAE of 1.715, and R^2 score of 0.9996. Thus, on average, the square of the residuals between the predicted and expected values of the response variable is 4.965. MSE is more sensitive to outliers and penalises them more. More insights into model performance can be determined from the MAE value of 1.715, which indicates that, on average, the absolute difference between the predicted and expected duration is 1.715 seconds. This small value is acceptable for the purposes of this study.

The value of the R^2 score determines the goodness of fit of our model. It measures how well the model fits the dataset. Smaller values indicate that the model is under-fitting the dataset, while values closer to one indicate a good model fit. In our case, we obtained an R^2 score of 0.9996, which indicates extremely good model fitting. This is expected because the duration and input features are related by an explicit linear function. However, this high value of R^2 is not possible with a real dataset for several compelling reasons. First, there are many factors that influence the service duration in real networks, which are not considered in this work. Second, service duration is usually not linearly dependent on these factors.

The trained model was saved and loaded during network simulation to predict the duration of each incoming call request. The network was simulated under different scenarios to thoroughly investigate the performance of the proposed admission control scheme. The next section details the simulation results and their analysis.

5.5 Simulation Results of the Proposed Scheme

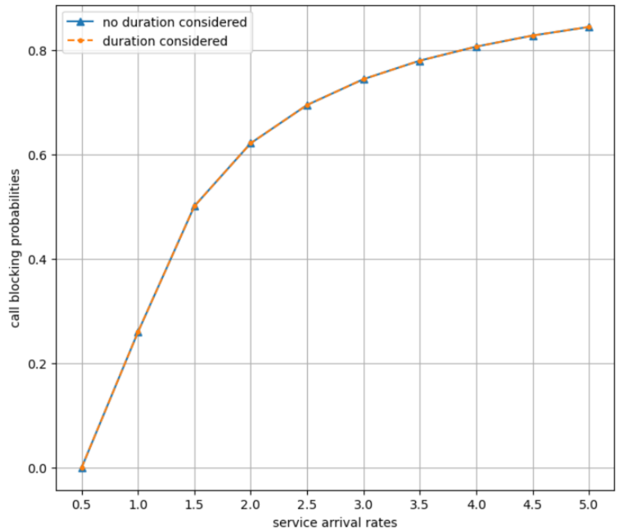
We conducted a series of experiments on the developed model to study the impact of service duration, user groups, and satellite visibility time on network performance. To demonstrate the importance of considering service duration in integrated terrestrial and non-terrestrial networks, we further used the developed model and compared it with the scenario when admission decisions did not consider service duration. The simulation results and their detailed analysis are presented in this section. It must be noted that these results were obtained by simulating the network for a significantly long time. Long enough for a steady state to be reached. This was so that the long-term effects of the random generators used in some parts of the simulation did not have significant effects on the expected results. For example, when sample user group type from Bernoulli distribution, to get 50% from each group, many samples needed to be sampled from the distribution. Additional results that were obtained by running the simulation for short simulation times are in the appendices.

5.5.1 Significance of Incorporating Service Duration

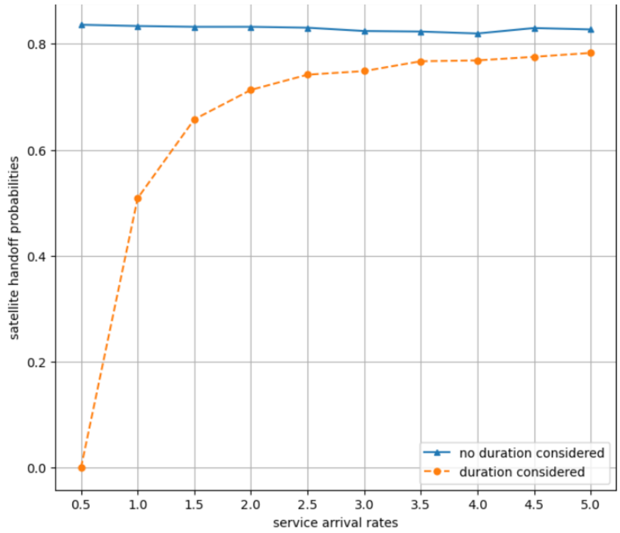
To demonstrate the importance of considering service duration in admission decisions, particularly for integrated terrestrial and non-terrestrial network, the network model was simulated under the following admission scenarios.

- **Scenario 1: Service Duration Not Considered.** In this scenario, the admission controller admits users into all the RATs as long as sufficient resources are available. No consideration is given to the service duration or its impact on network performance, such as handoff frequency.
- **Scenario 2: Service Duration Considered.** In this scenario, the admission control algorithm takes the predicted service duration into account, with the goal of reducing the frequency of handoff calls. Thus, this scenario uses the proposed predictive admission control algorithm.

For each scenario, the network consisted of user tat connected to all RATS (group A users), and service arrival rates were varied from 0.5 calls to 5 calls, in steps of 0.5 calls per second. The simulation was left to run for a substantial amount of time so that the effect of randomness on the results is reduced. For each arrival rate, we tracked the number of call attempts and number of calls that were blocked in each scenario. This was then used to calculate the call blocking probability for that particular arrival rate. Similarly for satellite handoff probability, we tracked the number of calls admitted in RAT-1 (Satellite Access Network) and the number of admitted calls that had their durations greater than satellite visibility times. The results of this investigation are shown in Figure 5.4.



(a) Effect of admission with and without duration prediction on call blocking probability



(b) effect of admission with and without duration prediction on satellite handoff probability

Figure 5.4: Effect of admission with and without duration prediction on network performance

Figure 5.4a illustrates the effect that the two admission control scenarios (with and without duration prediction) have on the call blocking probability. As depicted in the figure, there is no notable difference in the call blocking probability between the two scenarios. This is because the call blocking probability is mainly influenced by the network's resource limitations. When a call comes in and there are insufficient resources available to accommodate it, the call will be blocked, regardless of how long it is expected to last. Thus, duration prediction does not have a direct effect on the blocking decision in resource-constrained situations. This re-emphases the fact that, network resource optimization is still crucial in the next-generation networks, which will be highly dynamic and support extremely diverse services.

Moreover, it can be observed from Figure 5.4a that call blocking probability increases with increasing call arrival rates, as expected. At higher arrival rates, more call requests are made per unit time than the number of calls that are departing the network, leading to an increase in the call blocking probability. This behavior is typical in networks where the demand for resources exceeds the available supply at high traffic levels.

Figure 5.4b illustrates the effect of the two admission control scenarios on satellite handoff probability. As evident from the figure, when users are admitted without considering service duration, the network experiences significantly higher satellite handoff rates.The satellite handoff rate is so high that the minimum satellite handoff probability is no less than 0.8. This means, without considering service duration in admission decision, minimum above 80% of calls admitted in satellite will experience satellite handoffs. This outcome is undesirable in integrated terrestrial and non-terrestrial networks, particularly because high-speed satellite networks inherently trigger more frequent handoffs.

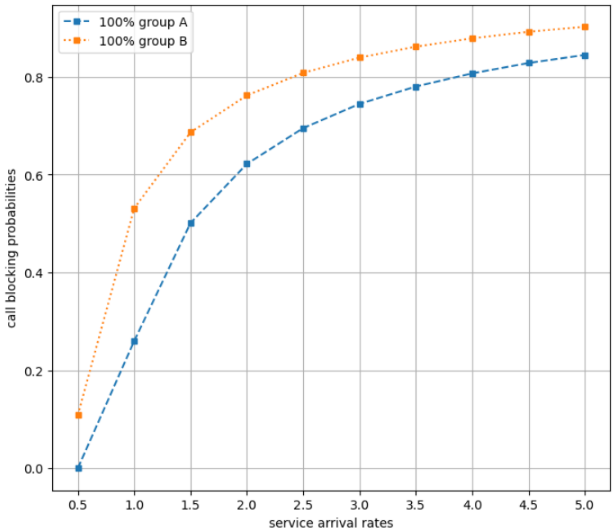
As demonstrated by Juan et al. [4], a higher handoff frequency leads to a greater likelihood of handoff failures (HFs), which can severely degrade service quality. Such degradation is especially critical in next-generation networks that aim to provide seamless connectivity across terrestrial and satellite systems. However, Figure 5.4b clearly shows that when service duration is predicted and taken into account during admission control, the frequency of satellite handoffs is reduced. In some instances, it is even possible to achieve zero satellite handoffs, as the predicted duration aligns with the satellite visibility time. This finding underscores the importance of incorporating service duration prediction in admission decisions to minimize handoffs and improve overall network performance in future integrated networks.

As illustrated in Figure 5.4b, satellite handoff rate increases with increasing arrival rates when calls are admitted using proposed admission control(considers duration). Meanwhile, the is less dependence of satellite handoff rates for admission control that does not consider service duration on arrival rates. This means that at all arrival rates- all traffic flows, the satellite handoff rates when duration is not considered are consistently high.

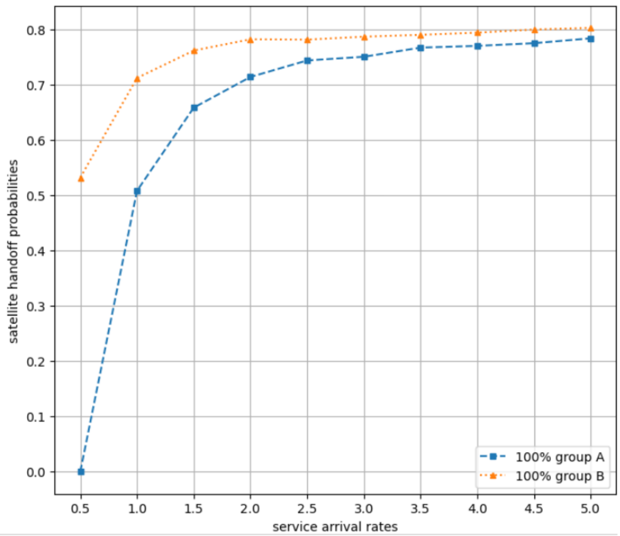
5.5.2 Effect of User Group on Network Performance

The network model considered in this work consists of two user groups: Group A users connect to all available RATs and Group B users who only connect to satellite access network(RAT-1),and terrestrial 5G macro base station(RAT-2). To study how different proportions of this user groups impact the performance of the proposed admission control scheme, simulation were run under different ratios of each group in the network.

We started by considering the impact of each group , individually , on call blocking and satellite handoff rates. The results are shown in Figure 5.5.



(a) Effect of individual user group on call blocking probability

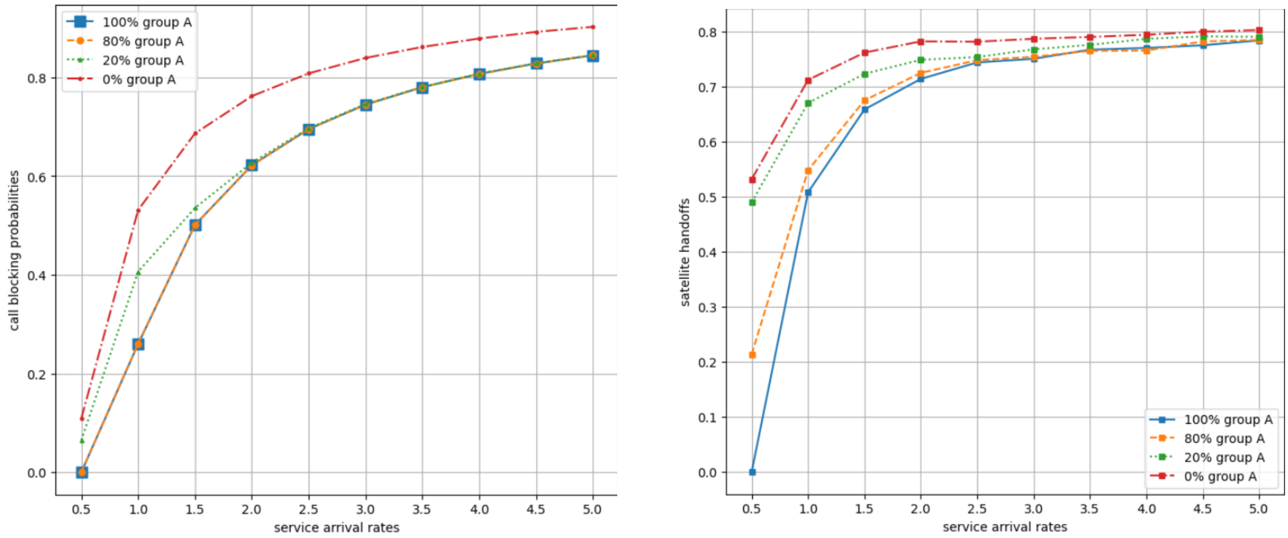


(b) effect of individual user group on satellite handoff probability

Figure 5.5: Effect of individual user group on network performance

We also consider scenarios when there are different proportions of these two user groups in the network. Specifically, we simulated the network with 20% and 80% of group A users. The results are shown in Figure 5.6.

Figures 5.5a and 5.5b illustrate how these groups individually impact call blocking and satellite handoff probabilities. As evident from Figure 5.5a, Group B users experience higher call blocking probabilities compared to Group A users. In some instance, the blocking rate of group B users is almost twice as much as that of group A calls. This is because Group A users have access to three radio access networks (RATs), while Group B users are limited to two. Consequently, there are technically more radio resources available for Group A users, leading to lower call blocking rates. These results highlight the importance of having multiple accessible RATs in mobile networks for reducing call blocking.



(a) Effect of mixed proportions of user groups on call blocking probability

(b) Effect of different proportions of user groups on satellite handoff probability

Figure 5.6: Effect of individual user group on network performance

These results are in agreement with findings made in the work in [66], which considered network selection for mixed deployments of 5G NSA and SA networks. It was noted therein that users whose equipment was limited to LTE suffered the highest blocking probabilities. This further strengthens the assertion that merely deploying multiple radio access networks is not enough. The users will also require devices that are capable of connecting to as many varied networks as possible for the full utilization of the available features. This is a problem likely to persist in next generation wireless networks given that a large number of subscribers are currently unable to afford 5G-enabled devices. In fact, users may still not be able to afford devices capable of connecting directly to satellite networks in the developing countries. Therefore, network operators should be offering a range of options in devices that support the advancing technologies at cheap prices. One probable solution might be to loan UEs to the subscribers or else consider cheaper options.

The results in Figure 5.5b indicate that users from group B experience significantly high satellite handoff rates when compared to group A. This is because of Group B's reliance on the satellite access network, which is subject to satellite visibility constraints, leading to more frequent handoffs. It can be observed from Figure 5.5b that the trend is that increase in arrival rates results in increase in satellite handoffs. This illustrate that at high traffic, the network may still experience high rates of satellite handoffs,despite the consideration of admission control. This further reiterates the fact that network operators still need to optimise their resource allocation is effectively serve large traffic.

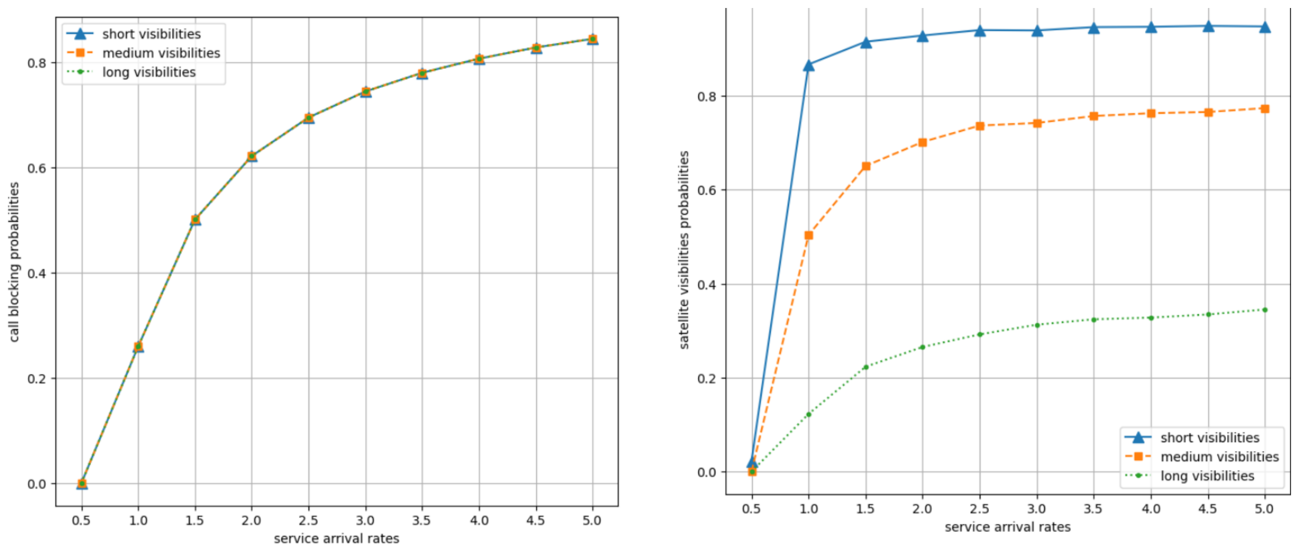
In Figure 5.6a, the effect of different proportions of user groups on call blocking probability is illustrated. It can be observed that when users in Group A are less than Group B users, the network has a high call-blocking probability, but as the number of users of Group A increases, the blocking probability decreases due to additional RATs available to them. Beyond a 20% share, the figure shows that this effect tails off, especially for larger arrival rates from 1.5 calls per second onward. At these rates, all RATs are completely saturated, and no further users can be fitted in regardless of Group A's share taken. This implies that capacity limits are inherent problem affecting network performance even in advanced networks such as integrated terrestrial and non-terrestrial systems. Hence, network operators should strive to highly optimize their resource allocation.

Figure 5.6b illustrates the effect that different proportions of group A and group B users have on satellite handoff probability. At first glance it can be observed that when there are 0% and 20% group A users, the network experiences significantly high satellite handoff probability as compared to when there are 80% and 100% of group A users. This is as expected because group A has extra terrestrial RAT that they can connect to when satellite is having short visibility times. The figure further shows that satellite handoff probability increases with increasing probability, as expected. This suggests that, at high arrival rates, the incoming traffic is so high that consideration of service duration in admission decision has no favourable effect.

5.5.3 Effect of Satellite Visibility Time on Network Performance

Since satellites at different altitudes have different orbital speeds, and thus different visibility times, we conducted an experiment to understand how different visibility times of satellites can influence the performance of the proposed admission decision.

The experiment involved simulating the network with different scenarios of satellite visibility times. We considered visibility times less than 200 seconds as *short visibilities*, visibilities between 200 and 400 seconds as *medium visibilities*, and those above 400 seconds as *long visibilities*. The experimental setup involved network traffic consisting of 100% of Group A users. The arrival rates were varied from 0.5 calls per second to 5 calls per second, with increments of 0.5 calls per second. As before, the simulation was run long enough for the long-term effects of random numbers not to affect the results. The results illustrating the effect of visibility time on call blocking probability are shown in Figure 5.7a, while those illustrating the effect of visibility time on satellite handoff are shown in Figure 5.7b



(a) Effect of satellite visibility time on call blocking probability

(b) Effect of satellite visibility time on satellite handoff probability

Figure 5.7: Effect of visibility time on network performance

As shown in Figure 5.7a, there is no notable difference in the call blocking probabilities of these visibility scenarios. This can be explained by the fact that call blocking probability mainly relies on the availability of network resources rather than the length of satellite visibility. Provided that there are enough resources in the network, calls can be processed regardless of how long the satellite is visible. Also, it can be noted from the figure that call blocking probability increases with service arrival rates as expected.

On the other hand, Figure 5.7b indicates that satellite visibility time significantly impacts satellite handoff probability. In cases with short visibility times, the network faces significantly higher satellite handoff probabilities compared to those with medium and long visibility periods. For a short visibility scenario, the satellite handoff probability is so high that they are more than 5 times probabilities for long visibility times. This is due to the fact that shorter visibility times mean the satellite is only within the user's coverage area for a brief period, increasing the chances that an active call will need to switch to a new satellite before it concludes. Conversely, with longer visibility times, the satellite can maintain an ongoing call for a more extended period, thereby decreasing the frequency of handoffs.

Low orbital altitudes of satellites are usually preferred because they offer low Round Trip Time (RTT), especially for services such as uRLLC. However, satellites orbiting at low altitudes tend to have much higher orbital speeds, hence short visibility times. Thus, in satellite communication design, a trade-off will need to be made between low latency and visibility time (hhence frequency of handoffs). While satellite visibility time does not directly influence call blocking probability, it is crucial in determining how often satellite handoffs occur. Networks with shorter satellite visibility times are more susceptible to frequent handoffs, which can compromise the quality of service (QoS) and raise the risk of handoff failures. Thus, optimising satellite visibility and handoff strategies is vital for maintaining seamless communication in integrated terrestrial and non-terrestrial networks.

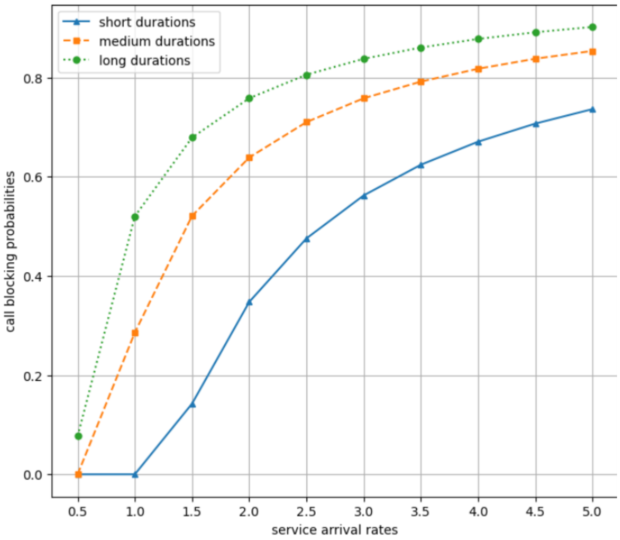
5.5.4 Effect of Duration on Network Performance

To investigate the impact that service duration has on network performance, three different scenarios of service duration were considered. Using the mean ($\mu = 418.04$ seconds) and standard deviation ($\sigma = 132.338$ seconds) of service duration as shown in Figure 5.2, we considered the following scenarios:

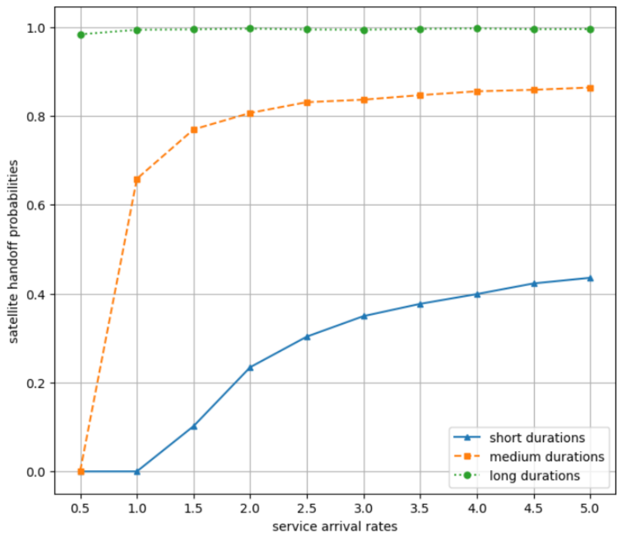
- **Short Calls:** Calls with a duration less than or equal to $\mu - \sigma = 285.702$ seconds were categorized as short calls.
- **Medium Calls:** Calls with a duration between $\mu - \sigma = 285.702$ seconds and $\mu + \sigma = 550.378$ seconds were categorized as medium-duration calls.
- **Long Calls:** Calls with a duration greater than or equal to $\mu + \sigma = 550.378$ seconds were categorized as long calls.

During the simulations, the total traffic was 100% from group A (users that connect to all RATs). The results of the simulation under each service duration scenario are presented in Figure 5.8. The effect of different service duration scenarios on call blocking probability is shown in Figure 5.8a, while Figure 5.8b shows the impact of service duration on satellite handoff probability.

Figure 5.8a illustrates how different scenarios of service duration influence call blocking probability. From the figure, it can be observed that longer calls tend to have higher call blocking probabilities than shorter ones. This occurs because long-duration calls use up radio resources for a longer time, which limits the availability of resources for new incoming calls, resulting in increased blocking rates. In contrast, short-duration calls free up network resources more quickly, leading to lower call blocking probabilities. As anticipated, the call blocking probability rises across all scenarios as service arrival rates increase, reflecting the greater demand for network resources during peak periods.



(a) Effect of different service duration scenarios on call blocking probability



(b) Effect of different service duration scenarios on satellite handoff probability

Figure 5.8: Effect of service duration on network performance

Figure 5.8b depicts the impact of service duration on satellite handoff probability. Long-duration calls show significantly higher satellite handoff probabilities compared to medium- and short-duration calls. Almost all long-duration calls admitted in satellite networks experience satellite handoffs. This is due to the fact that longer calls are more likely to exceed the satellite's visibility period, necessitating handoffs to maintain a connection. Conversely, shorter calls are more likely to finish before a satellite handoff is needed, resulting in fewer handoffs. In some instances, short calls did not experience satellite handoffs at all.

5.6 Conclusion

In this chapter, we presented the results of simulating the proposed admission control algorithm. It was the model training results that were presented and analysed. Next, we conducted a series of experiments on the proposed admission control to study different network scenarios, and all demonstrated that more care is needed during peak times, particularly for long-duration calls. The next chapter draws conclusions based on the findings in the chapter and makes recommendations based on these conclusions.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The integration of terrestrial and non-terrestrial networks to achieve global coverage presents the unavoidable challenge of frequent handoffs, particularly due to the high-speed nature of satellite systems. This project addressed this problem by developing a predictive admission control and bandwidth allocation scheme that used a Generalized Linear Model to predict service duration and incorporated these predictions into admission decisions. Based on simulation results, the following conclusions has been drawn.

6.1.1 Extremely High Accurate Prediction Model

The service duration prediction model in this study was extremely accurate, giving R^2 scores as high as 0.9999. This is as result of reliance on synthetic data. It can not be concluded from these results that the model will perform exceptional when trained on real dataset. Furthermore, very few features were used when training. This resulted in the predictive model not giving a realistic picture about how it would perform in real situations.

6.1.2 Reduced Satellite Handoffs in Duration-based Admission Control

When duration was not considered in the admission decision, simulation results revealed that more than 80% of calls admitted to the satellite network experienced satellite handoffs. This high satellite handoff probability is undesirable because high-speed satellite networks often result in handoff failures, which can lead to service degradation. However, when calls were admitted based on their predicted durations, the results showed that satellite handoffs could be completely avoided in some instances. The advantage of reducing the probability of satellite handoffs is that network operators would not need to deploy complex handover algorithms capable of handling both user and satellite mobility.

6.1.3 Significant Impact of User Groups on Network Performance

The network model in this work consisted of two groups of users: those who could connect to all RATs, and those who could connect to only some of them. When investigating the impact of these user groups on performance, the simulation results showed that users who could not connect to all RATs experienced high call blocking and satellite handoff rates. Meanwhile, users capable of connecting to all RATs experienced reduced call blocking and handoff rates. It was also found that there is a point beyond which increasing the number of users capable of connecting to all RATs had no further effect on satellite handoff and call blocking rates. The conclusion drawn from these findings is that, although having multiple RATs co-existing in densely populated regions can improve performance, it is equally important for users to upgrade their equipment to be able to connect to all RATs.

6.1.4 Notable Impact of Service Durations on Network Performance

The developed model was used to investigate the effect of duration on call blocking and satellite handoff rates. The results showed that long-duration calls suffered the highest call blocking and satellite handoff rates. In some cases, the call blocking probability for long calls was as high as 95%. It is clear from these findings that long durations can deteriorate network performance and therefore require extra consideration when admitting such calls. However, short durations experienced very low satellite handoff rates.

6.1.5 Profound Impact of Satellite Visibility Times on Network Performance

The research also used the developed model to study the effect of satellite visibility time on network performance. It was found that, although visibility time had no notable effect on call blocking probability, shorter visibility times resulted in significantly higher satellite handoffs. Satellites orbiting at lower orbits have advantage of short round-trip times, but their high speed imply that their visibility will be very short,resulting in high handoff rates. This findings suggest that the trade-off will need to be made between low latency and service degradation resulting for potential handoff failures. Should network operators choose to deploy satellite networks in much low orbits, below 1000 km, efficient handoff algorithms will need to be put in place.

6.2 Recommendations

On the basis of above conclusions, the following recommendations are made.

6.2.1 Train the Predictive Model on Real-Network Dataset

The inherent limitation of this work is the reliance on a synthetic network data to train and evaluate the model. Using synthetic data, real network scenarios could not be accurately captured by the proposed model. This implies that the model's performance in a real network environment remains uncertain. It is therefore recommended to use real telecommunication call detail record data to train the predictive model. Furthermore, service quality depends on several other factors, such as location, which were not considered in this work. Taking into account as many factors that influence service duration as possible could help in developing a predictive model that generalizes well to real network conditions.

6.2.2 Compute Real Satellite Visibility Time

The use of a random generator to model satellite visibility time limits the validity of the results of this study. It is therefore recommended that real visibility times be computed based on the user's location and minimum elevation angle. This would involve an in-depth understanding of orbital mechanics to accurately calculate the visibility of the satellite given the user's location. Furthermore, it is recommended to incorporate satellite orbital parameters such as altitude, inclination, and velocity into the visibility calculations. These parameters will allow for more precise modeling of satellite coverage areas and handoff patterns.

6.2.3 Consider User Requirements in Admission Decisions

In this study, admission decisions were made without considerations of user requirements. For instance, admission o uRLLC service on satellite just because its predicted duration is less that visibility time of the satellite might not be the most optimal way to go about it. Consequently, if is recommended that future works consider incorporation user requirements in admission decision, in addition to service duration. This will enhance user satisfaction, as so network's quality of service provision.

6.2.4 Develop Adaptive Predictive Model

The predictive model developed in this study was trained offline, and used during network simulation to predict user durations. This resulted in model making significant errors for caller-callee pairs that were not in the training dataset. It is therefore recommended that future works should consider developing adaptive models that continuously learn the dynamics of the network, in real-time. This will help the model to capture new emerging user calling patterns, and achieve more prediction accuracy.

Bibliography

[1] H. Yao, L. Wang, X. Wang, Z. Lu, and Y. Liu, "The space-terrestrial integrated network: An overview," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 178–185, 2018.

[2] M. O. Ojijo and O. E. Falowo, "A survey on slice admission control strategies and optimization schemes in 5g network," *IEEE Access*, vol. 8, pp. 14 977–14 990, 2020.

[3] X. You, C.-X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, J. Wang *et al.*, "Towards 6g wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, vol. 64, pp. 1–74, 2021.

[4] E. Juan, M. Lauridsen, J. Wigard, and P. E. Mogensen, "5g new radio mobility performance in leo-based non-terrestrial networks," in *2020 IEEE Globecom Workshops (GC Wkshps*. IEEE, 2020, pp. 1–6.

[5] 3GPP. Non-terrestrial networks (ntn). Accessed on: 10 October 2024. [Online]. Available: https://www.3gpp.org/technologies/ntn-overview

[6] E. S. Agency. 5g-goa:5g enabled ground segment technologies over the air demonstrator. Accessed on: 10 October 2024. [Online]. Available: https://connectivity.esa.int/projects/5ggoa

[7] ——. 5g-leo:openairinterface™ extension for 5g satellite links. Accessed on: 10 October 2024. [Online]. Available: https://connectivity.esa.int/projects/5gleo

[8] W. Jiang, *Cellular Communication Networks and Standards: The Evolution from 1G to 6G*. Springer Nature, 2024.

[9] S. Shukla, V. Khare, S. Garg, and P. Sharma, "Comparative study of 1g, 2g, 3g and 4g," *J. Eng. Comput. Appl. Sci*, vol. 2, no. 4, pp. 55–63, 2013.

[10] J. R. Churi, T. S. Surendran, S. A. Tigdi, and S. Yewale, "Evolution of networks (2g-5g)," in *International Conference on Advances in Communication and Computing Technologies (ICACACT)*, vol. 51, no. 4. Citeseer, 2012, pp. 8–13.

[11] N. Bhandari, S. Devra, and K. Singh, "Evolution of cellular network: from 1g to 5g," *International journal of engineering and techniques*, vol. 3, no. 5, pp. 98–105, 2017.

[12] A. A. A. Solyman and K. Yahya, "Evolution of wireless communication networks: from 1g to 6g and future perspective," *International journal of electrical and computer engineering*, vol. 12, no. 4, p. 3943, 2022.

[13] P. Sharma, "Evolution of mobile wireless communication networks-1g to 5g as well as future prospective of next generation communication network," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 8, pp. 47–53, 2013.

[14] A. A. Salih, S. Zeebaree, A. S. Abdulraheem, R. R. Zebari, M. Sadeeq, and O. M. Ahmed, “Evolution of mobile wireless communication to 5g revolution,” *Technology Reports of Kansai University*, vol. 62, no. 5, pp. 2139–2151, 2020.

[15] N. T. M. Lan, “Evolution of wireless technology: From 1g to 5g,” *Asian J. Appl. Sci. Technol.*, vol. 7, no. 04, pp. 68–73, 2023.

[16] T. S. Rappaport, *Wireless communications: principles and practice*. Cambridge University Press, 2024.

[17] L. J. Vora, “Evolution of mobile generation technology: 1g to 5g and review of upcoming wireless technology 5g,” *International journal of modern trends in engineering and research*, vol. 2, no. 10, pp. 281–290, 2015.

[18] E. Ezhilarasan and M. Dinakaran, “A review on mobile technologies: 3g, 4g and 5g,” in *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*, 2017, pp. 369–373.

[19] O. T. Eluwole, N. Udoh, M. Ojo, C. Okoro, and A. J. Akinyoade, “From 1g to 5g, what next?” *IAENG International Journal of Computer Science*, vol. 45, no. 3, 2018.

[20] O. Delgado and B. Jaumard, “Scheduling and resource allocation for multiclass services in lte uplink systems,” in *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, 2010, pp. 355–360.

[21] Z. Qadir, K. N. Le, N. Saeed, and H. S. Munawar, “Towards 6g internet of things: Recent advances, use cases, and open challenges,” *ICT express*, vol. 9, no. 3, pp. 296–312, 2023.

[22] N. Shaik and P. K. Malik, “A comprehensive survey 5g wireless communication systems: open issues, research challenges, channel estimation, multi carrier modulation and 5g applications,” *Multimedia Tools and Applications*, vol. 80, no. 19, pp. 28 789–28 827, 2021.

[23] S. Sharma, M. Deivakani, K. S. Reddy, A. Gnanasekar, and G. Aparna, “Key enabling technologies of 5g wireless mobile communication,” in *Journal of Physics: Conference Series*, vol. 1817, no. 1. IOP Publishing, 2021, p. 012003.

[24] E. Ivanova, T. Iliev, I. Stoyanov, and G. Mihaylov, “Evolution of mobile networks and seamless transition to 5g,” in *IOP Conference Series: Materials Science and Engineering*, vol. 1032, no. 1. IOP Publishing, 2021, p. 012008.

[25] S. Wijethilaka and M. Liyanage, “Survey on network slicing for internet of things realization in 5g networks,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[26] 3GPP. Overview of 5g systems. Accessed: Aug 1, 2024. [Online]. Available: https://www.3gpp.org/technologies/5g-system-overview

[27] R. F. Olimid and G. Nencioni, “5g network slicing: A security overview,” *Ieee Access*, vol. 8, pp. 99 999–100 009, 2020.

[28] M. Casoni, C. A. Grazia, M. Klapez, N. Patriciello, A. Amditis, and E. Sdongos, “Integration of satellite and lte for disaster recovery,” *IEEE Communications Magazine*, vol. 53, no. 3, pp. 47–53, 2015.

[29] Y. Qian, "Beyond 5g wireless communication technologies," *IEEE wireless communications*, vol. 29, no. 1, pp. 2–3, 2022.

[30] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6g wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.

[31] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6g networks: Use cases and technologies," *IEEE communications magazine*, vol. 58, no. 3, pp. 55–61, 2020.

[32] M. Rasti, S. K. Taskou, H. Tabassum, and E. Hossain, "Evolution toward 6g multi-band wireless networks: A resource management perspective," *IEEE Wireless Communications*, vol. 29, no. 4, pp. 118–125, 2022.

[33] C.-X. Wang, X. You, X. Gao, X. Zhu, Z. Li, C. Zhang, H. Wang, Y. Huang, Y. Chen, H. Haas *et al.*, "On the road to 6g: Visions, requirements, key technologies, and testbeds," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 905–974, 2023.

[34] M. Series, "Imt vision–framework and overall objectives of the future development of imt for 2020 and beyond," *Recommendation ITU*, vol. 2083, no. 0, pp. 1–21, 2015.

[35] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, "The road towards 6g: A comprehensive survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.

[36] M. Banafaa, I. Shayea, J. Din, M. H. Azmi, A. Alashbi, Y. I. Daradkeh, and A. Alhammadi, "6g mobile communication technology: Requirements, targets, applications, challenges, advantages, and opportunities," *Alexandria Engineering Journal*, vol. 64, pp. 245–274, 2023.

[37] T. Randhawa and R. Hardy, "Performance evaluation of bandwidth partitioning in broadband networks," in *ATM 2000. Proceedings of the IEEE Conference on High Performance Switching and Routing (Cat. No.00TH8485)*, 2000, pp. 411–418.

[38] T. Zhang, E. Van Den Berg, J. Chennikara, P. Agrawal, J.-C. Chen, and T. Kodama, "Local predictive resource reservation for handoff in multimedia wireless ip networks," *IEEE Journal on selected areas in Communications*, vol. 19, no. 10, pp. 1931–1941, 2001.

[39] R. Kulshrestha, A. Agarwal *et al.*, "An adaptive fractional guard channel based cac scheme for heterogeneous traffic in wireless cellular networks," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2019, pp. 1260–1264.

[40] M. Mandour, A. D. Elbayoumy, G. M. A. Hamid, and A. M. Abdelaziz, "Dynamic channel allocation scheme for handover calls in cellular networks," in *2020 3rd International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2020, pp. 457–461.

[41] K. Sanon and S. Joshi, "Preemptive mobile assisted and guard channel based handoff queuing scheme," in *2011 Annual IEEE India Conference*. IEEE, 2011, pp. 1–4.

[42] İ. Candan, "Mobility and queue based guard channel scheme for cellular networks," in *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*. IEEE, 2014, pp. 1–4.

[43] A. Alioua, N. Gharbi, and S.-M. Senouci, “Call admission control scheme using borrowable guard channels and prioritizing fresh calls retrials in small cell networks,” in *2014 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2014, pp. 1–6.

[44] O. E. Falowo and H. A. Chan, “Joint call admission control algorithms: Requirements, approaches, and design considerations,” *Computer Communications*, vol. 31, no. 6, pp. 1200–1217, 2008.

[45] P. Fazio, M. Tropea, C. Sottile, S. Marano, M. Voznak, and F. Strangis, “Mobility prediction in wireless cellular networks for the optimization of call admission control schemes,” in *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2014, pp. 1–5.

[46] F. Yu and V. Leung, “Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks,” *Computer Networks*, vol. 38, no. 5, pp. 577–589, 2002.

[47] S. Kumar, K. Kumar, and P. Kumar, “Mobility based call admission control and resource estimation in mobile multimedia networks using artificial neural networks,” in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*. IEEE, 2015, pp. 852–857.

[48] A. Belhadj, K. Akilal, S. Bouchelaghem, M. Omar, and S. Aissani, “Next-cell prediction with lstm based on vehicle mobility for 5g mc-iot slices,” *Telecommunication Systems*, pp. 1–25, 2024.

[49] S. Xiao and W. Chen, “Dynamic allocation of 5g transport network slice bandwidth based on lstm traffic prediction,” in *2018 IEEE 9th international conference on software engineering and service science (ICSESS)*. IEEE, 2018, pp. 735–739.

[50] V. Perifanis, N. Pavlidis, R.-A. Koutsiamanis, and P. S. Efraimidis, “Federated learning for 5g base station traffic forecasting,” *Computer Networks*, vol. 235, p. 109950, 2023.

[51] A. Azari, P. Papapetrou, S. Denic, and G. Peters, “User traffic prediction for proactive resource management: Learning-powered approaches,” in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[52] S. Xiao and W. Chen, “Dynamic allocation of 5g transport network slice bandwidth based on lstm traffic prediction,” in *2018 IEEE 9th international conference on software engineering and service science (ICSESS)*. IEEE, 2018, pp. 735–739.

[53] A. Azari, F. Salehi, P. Papapetrou, and C. Cavdar, “Energy and resource efficiency by user traffic prediction and classification in cellular networks,” *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 2, pp. 1082–1095, 2021.

[54] B. Bao, H. Yang, Q. Yao, L. Guan, J. Zhang, and M. Cheriet, “Resource allocation with edge-cloud collaborative traffic prediction in integrated radio and optical networks,” *IEEE Access*, vol. 11, pp. 7067–7077, 2023.

[55] Y. Kawamoto, M. Takahashi, S. Verma, N. Kato, H. Tsuji, and A. Miura, “Traffic prediction-based dynamic resource control strategy in haps-mounted mec-assisted satellite communication systems,” *IEEE Internet of Things Journal*, 2023.

[56] K. L. Dias, S. F. Fernandes, and D. F. Sadok, "Predictive call admission control for all-ip wireless and mobile networks," in *Proceedings of the 2003 IFIP/ACM Latin America conference on Towards a Latin American agenda for network research*, 2003, pp. 131–139.

[57] Pennstate. Arma/arima overview. Accessed on: 22 September 2024. [Online]. Available: https://www.e-education.psu.edu/meteo820/node/559#:~:text=An%20ARIMA%20model%20is%20an,the%20order%20of%20the%20differencing.

[58] S. Dubba, S. Gupta, and B. R. Killi, "Predictive resource allocation and vnf deployment using ensemble learning," *Multimedia Tools and Applications*, pp. 1–26, 2024.

[59] W. Binghui, N. V. Abhishek, P. Amogh, and M. Gurusamy, "Npra: A novel predictive resource allocation mechanism for next generation network slicing," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 2023, pp. 716–721.

[60] R. Wu, G. Zhu, X. Lu, and G. Ning, "A predictive call admission control algorithm for wireless/mobile networks," in *IEEE Vehicular Technology Conference*. IEEE, 2006, pp. 1–5.

[61] W. Lee and J. Park, "Llm-empowered resource allocation in wireless communications systems," *arXiv preprint arXiv:2408.02944*, 2024.

[62] Solutions for nr to support non-terrestrial networks (ntn) (release 16)," 3gpp, tr 38.821 v1.0.0, 2019. Accessed:20 September, 2024. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/38_series/38.811

[63] Base station (bs) radio transmission and reception (3gpp ts 38.104 version 16.4.0 release 16). Accessed:29 September,2024. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138100_138199/138104/16.04.00_60/ts_138104v160400p.pdf

[64] C. Bouras, G. Diles, V. Kokkinos, and A. Papazois, "Transmission optimizing on dense femtocell deployments in 5g," *International Journal of Communication Systems*, vol. 29, no. 16, pp. 2388–2402, 2016.

[65] M. Songailaite and T. Krilavicius, "Synthetic call detail records generator." in *IVUS*, 2021, pp. 102–111.

[66] O. Falowo, "Investigation of network selection decisions for mixed deployment of sa and nsa 5g network," 2024.

[67] S. Vhaduri, S. V. Dibbo, C.-Y. Chen, and C. Poellabauer, "Predicting next call duration: A future direction to promote mental health in the age of lockdown," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2021, pp. 804–811.

[68] E. W. Weisstein, "Beta distribution," *https://mathworld. wolfram. com/*, 2003.

[69] V. N. Ha, T. T. Nguyen, L. B. Le, and J.-F. Frigon, "Admission control and network slicing for multi-numerology 5g wireless networks," *IEEE Networking Letters*, vol. 2, no. 1, pp. 5–9, 2019.

[70] ZTE. Wi-fi 6 technology and evolution white paper. Accessed on: 23 September 2024. [Online]. Available: https://www.zte.com.cn/content/dam/zte-site/res-www-zte-com-cn/mediares/zte/files/pdf/white_book/Wi-Fi_6_Technology_and_Evolution_White_Paper-202009232125.pdf

[71] R. Udoh and U. Ukommi, "Determination of visibility time for geodesic satellites orbiting the earth on circular orbit subject to minimum zenith angle restriction," *International Multilingual Journal of Science and Technology (IMJST)*, vol. 7, no. 6, pp. 5435–5443, 2022.

[72] J. Neuhaus and C. McCulloch, "Generalized linear models," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 3, no. 5, pp. 407–413, 2011.

[73] S. Ahmed. A comprehensive introduction to generalized linear models. Accessed: Sep 8, 2024. [Online]. Available: https://medium.com/@sahin.samia/a-comprehensive-introduction-to-generalized-linear-models-fd773d460c1d

[74] R. A. Chumacero, "Ordinary least squares: A review," 2012.

[75] K. Lakshmi, B. Mahaboob, M. Rajaiah, and C. Narayana, "Ordinary least squares estimation of parameters of linear model," *J. Math. Comput. Sci.*, vol. 11, no. 2, pp. 2015–2030, 2021.

[76] V. Sampaio. Understanding ordinary least squares (ols): The foundation of linear regression. Accessed on: 9 October 2024. [Online]. Available: https://medium.com/@VitorCSampaio/understanding-ordinary-least-squares-ols-the-foundation-of-linear-regression-1d79bfc3ca35

[77] R. Agrawal. Know the best evaluation metrics for your regression model. Accessed on: 12 September 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

[78] S.-M. Senouci, A.-L. Beylot, and G. Pujolle, "Call admission control in cellular networks: a reinforcement learning solution," *International journal of network management*, vol. 14, no. 2, pp. 89–103, 2004.

[79] R. I. Vatasoiu, A. Vulpe, R. Florescu, M.-A. Sachian, and G. Suciu, "Developing a call detail record generator for cultural heritage preservation and theft mitigation: Applications and implications," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 2024, pp. 1–5.

[80] C.-D. Lai, D. Murthy, and M. Xie, "Weibull distributions and their applications," in *Springer Handbooks*. Springer, 2006, pp. 63–78.

[81] P. Biradar, P. Shweta, K. Padmanaban, A. Suresh, M. Bakhar, and A. Yeshitla, "Active learning assisted admission and bandwidth management in hwn for facilitating differential qos under multicriteria factors," *Scientific Programming*, vol. 2022, no. 1, p. 3175070, 2022.

[82] B. D. Embru Turgal. Include or exclude a constant term in regression analysis. Accessed on: 14 October 2024. [Online]. Available: https://www.researchgate.net/publication/317166884_Include_or_Exclude_a_Constant_Term_in_Regression_Analysis

Chapter 7

Appendix A

A1: Graduate Attributes Form

Table 7.1: Graduate Attribute forms

Graduate Attribute	**Justification**	**Section in the Report**
GA1: Problem-Solving.	The study address a problem of frequent handoffs In integrated terrestrial and non-terrestrial networks. I solved this problem by developing a predictive admission control Algorithm that Aims to reduce frequency of handoffs.	System Model: Chapter 3
GA4: Investigations, Experiments and Data Analysis	I reviewed the existing literature on call admission Control algorithms. I have implemented the Proposed model and conducted series of experiments to study its performance.	Lit Review: Chapter 2 Results and Analysis: Chapter5
GA5: Use of Engineering Tools	The proposed model was implemented using Python Libraries, and simulated on a Jupyter notebook. . Predictive model was trained using Ordinary Least Squares.	System Implementation: Chapter 4 Ordinary Least Squares: Chapter 3.3
GA6: Professional And technical communication	This report adheres of academic writing standards Good academic referencing styles has been used To credit the work of external authors.	Chapter 1-6
GA8: Individual work	The report is in my individual work. I individually Review the literature, developed the predictive Model and implemented it. Weekly supervisor have been held to track progress, and I Adhered to timeline of the project.	Chapter 1-5
GA9: Independent Learning Ability	I independently reviewed the literature and followed Proper referencing style. I also implemented the Proposed admission control algorithm independently	Chapter 2-6

A2: Ethics Clearance

PRE-SCREENING QUESTIONNAIRE OUTCOME LETTER

STU-EBE-2024-PSQ001232

2024/08/19

Dear Kananelo Chabeli,

Your Ethics pre-screening questionnaire (PSQ) has been evaluated by your departmental ethics representative. Based on the information supplied in your PSQ, it has been determined that you do not need to make a full ethics application for the research project in question.

You may proceed with your research project titled:

Predictive Admission Control and Bandwidth Allocation Scheme for Integrated Terrestrial and Non-Terrestrial Network

Please note that should aspect(s) of your current project change, you should submit a new PSQ in order to determine whether the changed aspects increase the ethical risks of your project. It may be the case that project changes could require a full ethics application and review process.

Regards,

Faculty Research Ethics Committee

Chapter 8

Appendix B

Imports

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from datetime import datetime,timedelta
import os
import random
import sys
import copy
#modules for Predictive Model implementation and Simulation
from sklearn.linear_model import LinearRegression
from sklearn.inspection import PartialDependenceDisplay
from sklearn.model_selection import train_test_split,KFold,cross_val_score
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
import statsmodels.api as sm
from statsmodels.tools import add_constant
from joblib import dump, load
from sklearn.preprocessing import PolynomialFeatures
import json

#Modules for plotting and analysis
import seaborn as sns
from statsmodels.distributions.empirical_distribution import ECDF

np.random.seed(42)
plt.rcParams['figure.figsize'] = [8, 7]
```

Simulation Parameters

```

#specify the date from which the data is to be generated: yyyy-mm-dd
start_date = '2024-01-01'

#Specify the end date which is the last date when the call detail records are generated: yyyy-mm-dd
```

```
end_date = '2024-01-31'

#Specify the number of customers in the network
customerNum = 100

#specify the filename where generated data will be saved to
filename = 'CDR.csv'

#specify the minimum number of contacts each user can have
min_contacts = 10

#specify the maximum number of contacts each user can have (mmust be less than 'customerNum')
max_contacts = 50

#load datafile if it already exists
if os.path.exists(filename) and os.path.exists('users.joblib'):
    load_file = True
else:
    load_file = False

#Specifty the time of the day when the simulation should start. It is crucial for predictive model
#must be in 24-hour format(e.g 2:30 pm must be specified ad 14.5)
start_hour = 6.0 #

#Specify number of hours for which simulations will ran for. (

#simulation result in report were produced with this set to 24, which takes a very long time to run
    te simulation
sim_time = 5.0/18.0

#specify list of arrival rates to simulate the network for.

arrival_rates = np.linspace(0.5,5,10)

#specify the number of BBUs required for a voice call

voice_bbu=1 #1 RB

#specify the capcities of each RAT

RAT_capacities = [52,#capaciy of RAT-1
                  133,# capacity of RAT-2
                  109 #city of RAT-3
                 ]
```

User Class

```

```

```python
#The user Class definition
class User:
    """
    Classs that implements the user in the network.
    """

    def  __init__(self,user_id,call_count=0):
        """
        Creates a new User object with the given ID, and initializes all
        parameters of the this User
        """
        self.ID =user_id

        #lits of contacts: users that this User can call, ove be called by this them
        self.contacts = []

        #distionary that stores relationship values of this User with its contacts
        self.relationships = {}

        #call log of all users this user called
        self.call_logs = {}

        #total calls the user can make per day
        self.call_count = call_count

        #flag to determine if the user is busy
        self.isBusy = False

    def add_contact(self,contact, relationship):
        """
        Adds the given contact into the calling user’s contact list and returns
        True if the addition was successful. Returns False otherwise(e.g if contact already exists)
        """
        if self.isContact(contact) or contact.ID == self.ID:
            return False
        else:
            self.contacts.append(contact)
            self.relationships[contact.ID] = relationship
            return True

    def isContact(self,user):
        """
        Retuns True if the given user is in this User’s contacts list.
        """
        for contact in self.contacts:
            if contact.ID  == user.ID:
                return True
        return False

```

```
    @staticmethod
    def get_time(peak_time=15.74, var = 4.62):
        """
        Returns the time (in 24-hour eg  13:30 is returned as 13.5) at which this user
        makes a call
        """
        return np.clip([np.random.normal(peak_time,var)],0,23.9999)[0]

    @staticmethod
    def adjust_time(time):
        """
        Returns the length of time between the given time and the peak time
        """
        return abs(time - 15.74)


    @staticmethod
    def get_one_time_duration(relationship,time):
        """
        returns the duration of the first call that this user makes with the contact of given
        relationship value.
        """

        duration = 30.75*relationship+7.75*User.adjust_time(time)


        return 0 if duration <0 else duration #eliminate negative duration

    @staticmethod
    def get_duration(relationship,time,avg_dur,noise_level=0.002):
        """
        Computes the duration of the current call that this user is about to make.
        """
        duration = 0.5*relationship+0.3*User.adjust_time(time)+0.95*avg_dur
        if duration <0:
            return 0
        else:
            return np.random.normal(duration,duration*noise_level)

def dial_call(self,time,full_time= None):
    """
    Select the user at random,and call them,if they are not busy( not in an on-going call)
    """
    #get contact IDs
    contact_IDs = np.array(list(self.relationships.keys()))

    #contact relationships(all of them)
    rels = np.array(list(self.relationships.values()))

```

```
    #make probabilities based on these relationships
    call_probs = rels/sum(rels)

    #select the user at random to call,with probability determined by
    callee_id = np.random.choice(contact_IDs,p=call_probs)

    #loop in the list of contact, to get the contact wit the selectd ID:
    for contact in self.contacts:
        if contact.ID == callee_id:
            callee = contact
            break
    #if this call is resampled, re-try another contact

    trials = np.random.randint (len(self.contacts)/2,len(self.contacts)) #random number of times to
        try calling contacts

    while (contact.isBusy and trials>0) or self.relationships[callee_id]==0:
        trials -= 1 #decrement trials.
        #try another contact
        callee_id = np.random.choice(contact_IDs,p=call_probs)

        for contact in self.contacts:
            if contact.ID == callee_id:
                callee = contact
                break
    if trials <= 0:
        return None #if the user couldn't make a call in given number of trials
    #other wise, the initiate the call to other user

    #get the relationship value
    rel= self.relationships[callee_id]

    #check to see if this call is the first call

    if callee.ID in self.call_logs:
        #if this call is not the firt call between the user and select contact,
        #procee it are follows:
        total_prev_calls = len(self.call_logs[callee.ID])
        #get current avg-duration
        avg_duration = self.call_logs[callee.ID][total_prev_calls-1][2]

        #get previous call duration
        dur_prev = self.call_logs[callee.ID][total_prev_calls-1][3]

        #get the number of previous calls made
        total_prev_calls = len(self.call_logs[callee.ID])

        #comuter new duration average(moving average formula)
        avg = avg_duration + (dur_prev-avg_duration)/(total_prev_calls)
```

```

        #get the actual duration that this call will last
        dur=User.get_duration(rel,time,avg)

        #build the call log
        call_log = [time,'voice',avg, dur]

        if full_time is not None:
            call_log.append(full_time)

        #add the call log to list of all call-logs made to the selected callee
        self.call_logs[callee.ID].append(call_log)
    else:
        #else, if this is the first calls, then process it as follows:

        #get duration using get_ont_time_duration
        dur = User.get_one_time_duration(rel,time)

        #build the first call log between this caller
        avg = 0
        call_log = [time, 'voice', avg, dur]

        if full_time is not None:
            call_log.append(full_time)

        #add this new call log in the list of call logs
        self.call_logs[callee.ID]=([call_log])

    #to show that the call is inprogress, set isBusy flag of this User and callee to True
    #and return the callee
    self.isBusy  = True
    callee.isBusy = True
    return callee,dur,avg

User.dial_call = dial_call

def dial_call_v1(self,time,full_time= None, generator=None):
    """
    Select the user at random,and call them,if they are not busy( not in an on-going call)
    """
    #get contact IDs
    contact_IDs = np.array(list(self.relationships.keys()))

    #contact relationships(all of them)
    rels = np.array(list(self.relationships.values()))

    #make probabilities based on these relationships
    call_probs = rels/sum(rels)

```

```
    #select the user at random to call,with probability determined by
    if generator is not None:
        callee_id = generator.choice(contact_IDs,p=call_probs)
    else:
        callee_id = np.random.choice(contact_IDs,p=call_probs)

    #loop in the list of contact, to get the contact wit the selectd ID:
    for contact in self.contacts:
        if contact.ID == callee_id:
            callee = contact
            break

    #get the relationship value
    rel= self.relationships[callee_id]

    #check to see if this call is the first call

    if callee.ID in self.call_logs:
        #if this call is not the firt call between the user and select contact,
        #procee it are follows:
        total_prev_calls = len(self.call_logs[callee.ID])
        #get current avg-duration
        avg_duration = self.call_logs[callee.ID][total_prev_calls-1][2]

        #get previous call duration
        dur_prev = self.call_logs[callee.ID][total_prev_calls-1][3]

        #get the number of previous calls made
        total_prev_calls = len(self.call_logs[callee.ID])

        #comuter new duration average(moving average formula)
        avg = avg_duration + (dur_prev-avg_duration)/(total_prev_calls)

        #get the actual duration that this call will last
        dur=User.get_duration(rel,time,avg)

        #build the call log
        call_log = [time,'voice',avg, dur]

        if full_time is not None:
            call_log.append(full_time)

        #add the call log to list of all call-logs made to the selected callee
        self.call_logs[callee.ID].append(call_log)
    else:
        #else, if this is the first calls, then process it as follows:

        #get duration using get_ont_time_duration
        dur = User.get_one_time_duration(rel,time)
```

```python

        #build the first call log between this caller
        avg = 0
        call_log = [time, 'voice', avg, dur]

        if full_time is not None:
            call_log.append(full_time)

        #add this new call log in the list of call logs
        self.call_logs[callee.ID]=([call_log])

    #to show that the call is inprogress, set isBusy flag of this User and callee to True
    #and return the callee
    return callee,dur,avg

User.dial_call_v1 = dial_call_v1
```

DataGenerator Class

```python

#The DataGenerator Class
class DataGenerator:
    """
    This class descibes an object that implements the generates synthetic data
    generation.
    """

    def __init__(self,N=150):

        #store the number of users to generate data for
        self.num_users = N

        #create array of users
        self.users = []

        #array of userr IDs
        self.user_ids = np.arange(1,N+1)
        #create users
        self.users = [User(idx,int(np.random.weibull(0.74) * 11.13)) for idx in self.user_ids]
        #flag that determines if the network of users is already built

        self.R = np.random.normal(10,5,size=(N,N))
        self.R = np.rint(self.R).astype(int)

        #remove personal calling possibilities
        np.fill_diagonal(self.R,0)

        #set relationshi values less than 0 to 0 ( to prevent negative duration)
```

```python
        self.R[self.R<0]= 0
        #network built flag
        self.network_built=False




    def build_network(self,min_contacts,max_contacts,verbose=False):
        """
        Build the network of users, assigning different users contacts.
        """
        #get the maximum number of contacts th
        self.network_built = True
        if verbose:
            print('***********Assigning Users contacts**************')

        for user in self.users:
            if verbose:
                print(f'\n\tAdding contacts for user-{user.ID}:',flush=True)
            friend_count = np.random.randint(min_contacts, max_contacts)
            for _ in range(friend_count):
                contact= np.random.choice(self.users) #get the user at random
                rel_caller = self.R[user.ID-1][contact.ID-1]


                if verbose:
                    print(f"\t\tTry adding contact with ID-{contact.ID}...",flush=True,end='')

                #resample again if the user already there
                while not user.add_contact(contact,rel_caller) or contact.ID == user.ID:

                    contact= np.random.choice(self.users)

                    if verbose:
                        print(f'Failed.\n\t\tTry adding contact with ID-{contact.ID}...',flush =
                            True,end='')

                if verbose:
                    print('Done.')

                #if the user not there, it should have been added in contact list
                #add user also in contact's contact list,if not there
                contact.add_contact(user,self.R[contact.ID-1][user.ID-1])

                #check to see the number of contacts already exceeded
                if len(user.contacts)> friend_count:
                    break #break and move to the next contact
    def reset(self):
        for user in self.users:
```

```
            user.isBusy = False

def generate_call_logs(self,start_date = '2024-01-01',end_date='2024-02-01',verbose=False):
    """
    Iterates through all days and hours of the given call and generate call logs.
    """

    if verbose:
        print('************Generating Call logs**************',flush=True)
    start_date_str = start_date.split('-')
    end_date_str = end_date.split('-')

        #converting this start date into datetime object.
    start = datetime(int(start_date_str[0]), int(start_date_str[1]), int(start_date_str[2]), 0, 0)
    end = datetime(int(end_date_str[0]), int(end_date_str[1]), int(end_date_str[2]), 0, 0)
    busy_users = [] #store users that are busy as we iterate through

    #start generate at 00:00:00 of the given start date
    current_time = start

    rates = [0.002,0.003,0.004,0.03,0.05] #random arrival rates

    np.random.seed(42) #ensure reproducibility

    while current_time <= end:

        timestep = np.random.exponential(1/np.random.choice(rates))

        #adjust the current time
        current_time+= timedelta(seconds=timestep)

        if current_time > end:
            break

        if verbose:
            print(f'\nCurrent Simulation Time:{str(current_time)}',flush=True)

        #release callers that have ended thier calls,upto this point
        for call in busy_users:
            if call[2] < current_time:
                call[0].isBusy = False
                call[1].isBusy = False
                busy_users.remove(call)
                if verbose:
                    print(f'Released caller-{call[0].ID} and callee-{call[1].ID}.',flush = True)

        #get anotherc aller
        caller = np.random.choice(self.users)
        if verbose:
```

```python
            print(f'Selected Caller-{caller.ID}. Checking if busy...',end='',flush=True)

        #select caller that is not busy for certain number of trials
        trials = np.random.randint(50,100)
        while caller.isBusy and trials > 0:
            trials -=1
            if verbose:
                print(f'Done.\nCaller was found busy. selecting another caller...',end='',flush=True)
            caller = np.random.choice(self.users) # get a caller ta random
            if verbose:
                print(f'Done.\nSelected Caller-{caller.ID}. Checking if busy...',end='',flush=True)
        if trials<=0:
            if verbose:
                print(f"Couldn't initiate a call. Operation aborted.")
            continue

        if verbose:
            print(f'Caller-{caller.ID} not busy. Initiating a call...',end= '', flush = True)

        #convert the current time to hours
        time = current_time.hour + current_time.minute / 60 + current_time.second / 3600

        call = caller.dial_call(time,current_time) #initiate a call
        if call is None:
            if verbose:
                print(f"Done.\nCouldn't initiate a call. Operation aborted.",flush=True)
            continue
        call_end = current_time + timedelta(seconds=call[1])
        busy_users.append((caller,call[0],call_end))
        if verbose:
            print('Done.\n\nCall Details:',flush=True)
            print(f'Caller-{caller.ID}.\nCalle-{call[0].ID}\nDuration:{round(call[1],3)} seconds.',
                flush=True)

    #save the network of users, with thier existing call logs to the file
    dump(self.users,'users.joblib')



DataGenerator.generate_call_logs = generate_call_logs

def extract_call_logs(self,verbose=False, filename = 'cdr.csv'):
    """
    Extracts call logs of users in the network, and logs them in readable way to external file
    """
    data = []
    for user in self.users:
        if verbose:
```

```
            print(f'Extracting call logs of user-{user.ID}...',end="",flush=True)
        for contact in user.contacts:
            if contact.ID in user.call_logs:
                if verbose:
                    print(f'\n\tExtracting call details with contact-{contact.ID}...',flush=True)
                i= 0
                for call_log in user.call_logs[contact.ID]:
                    if verbose:
                        print(f'call-log-{i+1}: {call_log}')
                    rel = self.R[user.ID-1,contact.ID-1]
                    data.append({
                        'caller-ID': user.ID,
                        'receiver-ID': contact.ID,
                        'relationship': rel,
                        'call-type': call_log[1],
                        'timestamp': call_log[4],
                        'adjusted-time(h)': round(User.adjust_time(call_log[0]),5),
                        'avg-duration(sec)': round(call_log[2],5),
                        'duration(sec)': round(call_log[3],5)
                    })
        if verbose:
            print('Done')
    np.random.shuffle(data)
    self.dataset= pd.DataFrame(data)
    #sort the file according to call time
    self.dataset=self.dataset.sort_values(by='timestamp',ascending=True)
    #save to file
    self.dataset.to_csv(filename,index=False)

DataGenerator.extract_call_logs = extract_call_logs

#create DataGenerator object with specified number of customers/users.
#the constructor takes the number of customers as the argument.

generator = DataGenerator(customerNum)

#if load-file set to True, load the new file, otherwise, generate new data
if load_file:

    #filename should be specifiedin the parameters sections
    generator.dataset = pd.read_csv(filename)

    #load the user's call log data used when generating the dataset
    generator.users = load('users.joblib')
else:
    #if load-file is not set to True, generate new dataset in the following steps:

    #step 1: Start by building the customer network

```

```
        #The function takes the minimum and maximum number of contacts each user can have

        #set the Verbose=True, if need to see the execution progress of the function
    generator.build_network(min_contacts = min_contacts, max_contacts = max_contacts,verbose=False)

    #step 2: generate call logs among users.

        #The function takes the start and end dates which define the period for which data should be
             generated

        #Set verbose=True if want to see the execution progress of the function

        #This function will generate a new file called 'users.joblib' which is a binary file that
            stores all
        #users details, such as thier contacts, and relationship status,as well as thier call logs.
    generator.generate_call_logs(start_date=start_date, end_date=end_date, verbose=False)

    #step 3: Extract the call logs of each user and dump them to the external file.
        #The function takes in the 'filename' parameter specified in parameters section and
        #Set verbose = True, is want to see functions execution status.
    generator.extract_call_logs(filename=filename,verbose=False)
```

Generating Synthetic Dataset

```
#create DataGenerator object with specified number of customers/users.
#the constructor takes the number of customers as the argument.

generator = DataGenerator(customerNum)

#if load-file set to True, load the new file, otherwise, generate new data
if load_file:

    #filename should be specifiedin the parameters sections
    generator.dataset = pd.read_csv(filename)

    #load the user's call log data used when generating the dataset
    generator.users = load('users.joblib')
else:
    #if load-file is not set to True, generate new dataset in the following steps:

    #step 1: Start by building the customer network

        #The function takes the minimum and maximum number of contacts each user can have

        #set the Verbose=True, if need to see the execution progress of the function
    generator.build_network(min_contacts = min_contacts, max_contacts = max_contacts,verbose=False)

    #step 2: generate call logs among users.
```

```
        #The function takes the start and end dates which define the period for which data should be
             generated

        #Set verbose=True if want to see the execution progress of the function

        #This function will generate a new file called 'users.joblib' which is a binary file that
            stores all
        #users details, such as thier contacts, and relationship status,as well as thier call logs.
    generator.generate_call_logs(start_date=start_date, end_date=end_date, verbose=False)

    #step 3: Extract the call logs of each user and dump them to the external file.
        #The function takes in the 'filename' parameter specified in parameters section and
        #Set verbose = True, is want to see functions execution status.
    generator.extract_call_logs(filename=filename,verbose=False)

dataset = generator.dataset

#sort the dataset in ascending order of timestampp
dataset = dataset.sort_values(by='timestamp',ascending=True)

#display first and last 5 samples of the dataset
dataset

#view statistics of the dataset
dataset.describe()

# Ensure 'durations' is a 1D array and numeric
durations = dataset['duration(sec)']

#plot the histogram of service durations
plt.hist(durations, bins=30, edgecolor='black', alpha=0.6)
plt.xlabel('Duration (seconds)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

Predictor Class

```

#Predictor Class
class Predictor:
    """
    Implemenets the service duration predictor object
    """

    def __init__(self,generator=None):
        self.model = None
```

```
        self.scaler = None
        self.dataGenerator = generator

    def train(self,dataset=None,verbose=False, output_filename = None):
        """
        Builds Generalized Linear Model, as well as testing it
        """
        if verbose:
            print('preprocessing data...',end='',flush=True)
        #do a little bit of filtering to remove outliers
        if dataset is None:
            dataset = self.dataGenerator.dataset
        data_filtered = dataset[(30<dataset['duration(sec)']) & (800>dataset['duration(sec)']) & (
            dataset['avg-duration(sec)']>30)]

        #shufle the dataset
        data_filtered = data_filtered.sample(frac=1, random_state=42).reset_index(drop=True)

        #separate inputs and output
        X = data_filtered.drop(['caller-ID', 'call-type','receiver-ID', 'timestamp', 'duration(sec)'
            ], axis=1)

        #the response variable
        y =data_filtered['duration(sec)']

        #split data into training and testing
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,shuffle=True)

        #scale features using MinMaxScaler
        self.scaler = StandardScaler()
        X_train_scaled = self.scaler.fit_transform(X_train)
        X_test_scaled = self.scaler.transform(X_test)
        if verbose:
            print('Done.\nTraining the Model...',flush=True,end='')
        ##add constant for OLS model
        X_train_const = sm.add_constant(X_train_scaled)
        X_test_const = sm.add_constant(X_test_scaled)

        self.model = sm.OLS(y_train,X_train_const).fit()

        if verbose:
            print('Done.\nModel Summary:')
            conf_intervals = self.model.conf_int()
            coefficients = self.model.params
            standard_errors = self.model.bse
            p_values = self.model.pvalues

            summary_df = pd.DataFrame({
                'Coefficients': coefficients,
```

```
                'SE': standard_errors,
                'CI_lower': conf_intervals[0],
                'CI_upper': conf_intervals[1],
                'p-value': p_values
            })

            print(summary_df)
            y_train_pred = self.model.predict(X_train_const)
            mse = mean_squared_error(y_train,y_train_pred)
            mae= mean_absolute_error(y_train,y_train_pred)
            r2= r2_score(y_train,y_train_pred)


        self.test_dataset = (X_test_const,y_test)
        if output_filename is not None:
            dump({"model": self.model, "scaler": self.scaler},output_filename)

    def evaluate(self,verbose=False):
        """
        Evaluates the predictor and returns MSE,MAE  and  R^2 scores
        """
        y_pred = self.model.predict(self.test_dataset[0])

        mse=mean_squared_error(self.test_dataset[1],y_pred)
        mae=mean_absolute_error(self.test_dataset[1],y_pred)
        r2=r2_score(self.test_dataset[1],y_pred)

        if verbose:
            print(f"Mean Squared Error:\t{round(mse,6)}")
            print(f"Mean Absolute Error:\t{round(mae,6)}")
            print(f"R-Squared Score:\t{round(r2,6)}")
        return mse,mae,r2

    def predict(self,input_vars):
        """
        predicts for a single input vector
        """
        feature_names = ['relationship', 'adjusted-time(h)','avg-duration(sec)']
        user_data_df = pd.DataFrame([input_vars], columns=feature_names)
        scaled_user_data = self.scaler.transform(user_data_df)
        scaled_user_data = np.column_stack((np.ones(scaled_user_data.shape[0]), scaled_user_data))
        duration = self.model.predict(scaled_user_data)
        return round(duration[0],6)

#predictor object

#Create a new predictor object. The constructor takes DataGenerator object as the argument
predictor = Predictor(generator)

```

```
#train the model.
#set the verbose=True, if want to see training results

predictor.train(verbose=True)

#evaluate the model. Set verbose to True if want to see evaluation results
eval_res=predictor.evaluate(verbose=True);
```

Simulator Class

```

#Simulator Class

class Simulator:
    """
    Network discrete simulation environemt
    """

    #voice resources
    voice_RB = voice_bbu
    rates = arrival_rates
    def __init__(self,predictor=None):
        """
        Creates new instance of the Simulator object

        """
        #keeps track to the total calls from each group that experience handoff
        if predictor is None:
            datagen = DataGenerator()
            datagen.dataset = pd.read_csv('cdr.csv')
            datagen.users = load('users.joblib')
            self.predictor = Predictor(datagen)
            self.predictor.train()
        else:
            self.predictor = predictor
    def sim_reset(self,test= None):
        """
        resets the simulation
        """
        self.handoffs = [0,0]
        self.blocked = [0,0]
        self.attempts = [0,0]
        self.handoffs_t = [0,0]
        self.rat_1_calls = [0,0]
        self.RATs=[RAT_capacities[0],RAT_capacities[1],RAT_capacities[2]]
        #load the original users, for each state
        self.predictor.dataGenerator.users = load('users.joblib')
        if test == 'prediction':
```

```
            #create parameters for prediction invesitgation
            self.handoffs_test = [0,0]
            self.blocked_test = [0,0]
            self.attempts_test = [0,0]
            self.handoffs_t_test = [0,0]
            self.rat_1_calls_test = [0,0]
            self.RATs_test=[RAT_capacities[0],RAT_capacities[1],RAT_capacities[2]]
            self.active_calls_test = []


def admit_no_duration(self,dur,vis,group):
    """
    Admits users without considering duration
    """
    if group == 'A':
        self.attempts_test[0]+=1
        if Simulator.voice_RB <= self.RATs_test[0]:
            self.RATs_test[0]-=Simulator.voice_RB
            self.rat_1_calls_test[0]+=1
            if dur > vis:
                self.handoffs_test[0]+=1
            return 1
        elif Simulator.voice_RB <= self.RATs_test[1]:
            self.RATs_test[1]-=Simulator.voice_RB
            return 2
        elif Simulator.voice_RB <= self.RATs_test[2]:
            self.RATs_test[2]-=Simulator.voice_RB
            return 3
        else:
            self.blocked_test[0]+=1
            return -1
    else:
        self.attempts_test[1]+=1
        if Simulator.voice_RB <= self.RATs_test[0]:
            self.RATs_test[0]-=Simulator.voice_RB
            self.rat_1_calls_test[1]+=1
            if dur > vis:
                self.handoffs_test[1]+=1
            return 1
        elif Simulator.voice_RB <= self.RATs_test[1]:
            self.RATs_test[1]-=Simulator.voice_RB
            return 2
        else:
            self.blocked_test[1]+=1
            return -1


Simulator.admit_no_duration= admit_no_duration

```

```python
def admission_control(self,dur,vis,group,actual_dur):
        """
        implements the admission control algorithm
        """
        if group == 'A':
            self.attempts[0]+=1
            if dur <= vis:
                if Simulator.voice_RB <= self.RATs[0]:
                    self.RATs[0]-=Simulator.voice_RB
                    self.rat_1_calls[0]+=1
                    if actual_dur > vis:
                        self.handoffs_t[0]+=1

                    return 1

                elif Simulator.voice_RB <= self.RATs[1]:
                    self.RATs[1]-=Simulator.voice_RB
                    return 2
                elif Simulator.voice_RB <= self.RATs[2]:
                    self.RATs[2]-=Simulator.voice_RB
                    return 3
                else:
                    self.blocked[0]+=1
                    return -1
            else:
                if Simulator.voice_RB <=self.RATs[1]:
                    self.RATs[1]-=Simulator.voice_RB
                    return 2
                elif Simulator.voice_RB <=self.RATs[2]:
                    self.RATs[2]-=Simulator.voice_RB
                    return 3
                elif Simulator.voice_RB <= self.RATs[0]:
                    self.RATs[0]-=Simulator.voice_RB
                    self.rat_1_calls[0]+=1
                    self.handoffs[0]+=1
                    return 1
                else:
                    self.blocked[0]+=1
                    return -1

        elif group == 'B':
            self.attempts[1]+=1
            if dur <= vis:
                if Simulator.voice_RB <= self.RATs[0]:
                    self.RATs[0]-=Simulator.voice_RB
                    self.rat_1_calls[1]+=1
                    #handoffs as result of predictive error
                    if actual_dur > vis:
                        self.handoffs_t[1]+=1
```

```python
                    return 1
                elif Simulator.voice_RB <= self.RATs[1]:
                    self.RATs[1]-=Simulator.voice_RB
                    return 2
                else:
                    self.blocked[1]+=1
                    return -1
            else:
                if Simulator.voice_RB <=self.RATs[1]:
                    self.RATs[1]-=Simulator.voice_RB
                    return 2
                elif Simulator.voice_RB <= self.RATs[0]:
                    self.RATs[0]-=Simulator.voice_RB
                    self.rat_1_calls[1]+=1
                    self.handoffs[1]+=1
                    return 1
                else:
                    self.blocked[1]+=1
                    return -1
        else:
            raise ValueError(f'Unsupported group:{group}')





Simulator.admission_control = admission_control

status = """
print(f'RAT Capacities After Admission:\\n\\tRAT-1:{self.RATs[0]}\\n\\tRAT-2:{self.RATs[1]}\\n\\tRAT
    -3:{self.RATs[2]}',flush=True)
if RAT>0:
    print(f'Admitted in RAT-{RAT}',flush=True)
    print(f'Admitted Call Details:\\n\\tCaller-ID:{caller.ID}\\n\\tCallee-ID:{callee.ID}',flush=True
        )
    print(f'\\tCaller Group:{grp}',flush=True)
    print(f'\\tNumber of calls between users:{len(caller.call_logs[callee.ID])} calls',flush=True)
    print(f'\\tAverage call duration of previous calls:{round(avg,3)} seconds',flush=True)
    print(f'\\tActual Call Duration:{round(dur,3)} seconds',flush=True)
    print(f'\\tPredicted Duration:{round(pred_dur,3)} seconds',flush=True)
    print(f'\\tVisibility Time:{round(vis,3)} seconds',flush=True)
else:
    print('Call Blocked.',flush=True)

print(f'\\nNetwork Parameters:\\n\\tNumber of call Attempts:\\n\\t\\tGroup A calls:{self.attempts
    [0]}',flush=True)
print(f'\\t\\tGroup B calls:{self.attempts[1]}',flush=True)
print(f'\\tNumber of blocked calls:\\n\\t\\tGroup A blocks:{self.blocked[0]}',flush=True)
print(f'\\t\\tGroup B blocks:{self.blocked[1]}',flush=True)
```

```
print(f'\\tNumber of satellite handoffs:\\n\\t\\tGroup A:{self.handoffs[0]}\\n\\t\\tGoup B:{self.
    handoffs[1]}',flush=True)
print(f'\\tNumber of calls in Satellite NW:\\n\\t\\tGroup A:{self.rat_1_calls[0]}',flush=True)
print(f'\\t\\tGroup B:{self.rat_1_calls[1]}',flush=True)
"""



pre_admin = """
print(f'RAT Capacities before Admission:\\n\\tRAT-1:{self.RATs[0]}\\n\\tRAT-2:{self.RATs[1]}\\n\\
    tRAT-3:{self.RATs[2]}',flush=True)

"""

def simulate(self,start_time=start_hour, total_time=sim_time,arrival_rate=0.5,groupA=1.0,test=[None,
    None],verbose=False):
    """
    Simulates the Network.

    """

    #generator that selects the caller
    caller_gen = np.random.default_rng(seed=48)

    #default random generator
    np.random.seed(42)
    #random generator to control visibility times independently

    vis_rng= np.random.default_rng(seed=40)

    #random generator to call arrivals (controlled independently)
    time_rng =  np.random.default_rng(seed=43)

    #random generator to select users
    user_rgn =np.random.default_rng(seed=44)

    #random generator that controls, user groop selection, independently
    group_rng = np.random.default_rng(seed=45)

    #start time
    start = timedelta(hours =start_time) #time to start the simulation

    #time to end the simulation
    end = start + timedelta(hours = total_time)

    #set the current time to start time
    current_time = start

    #list of active calls
```

```
    active_calls= []

    #reset all simulation parameters
    self.sim_reset(test =test[0])

    if test[0] == ’duration’:
        mean = self.predictor.dataGenerator.dataset[’duration(sec)’].mean()
        std = self.predictor.dataGenerator.dataset[’duration(sec)’].std()


    while current_time <= end:
        #generate call inter-arrival rate
        time_interval = time_rng.exponential(1/arrival_rate)

        #update the current time
        current_time += timedelta(seconds = time_interval)
        if verbose:
            print(f’\nCurrent Time:\t{str(current_time)}’,flush=True)
            exec(pre_admin,globals(),locals())
            print(f’\nChecking calls that have completed...’,flush=True,end=’’)
            i = 0
        #remove calls that ended before the current time
        for call in active_calls:

            if call[3] < current_time:

                call[0].isBusy = False #free the caller
                call[1].isBusy = False #free the callee
                self.RATs[call[2]-1]+=Simulator.voice_RB
                active_calls.remove(call)
                if verbose:
                    i+=1
                    print(f’done.\nEnded Call Details:\n\tCaller-{call[0].ID}\n\tCallee-{call[1].ID}\n\tRAT-{call[2]}’,flush=True)
        if verbose:
            print(’done.\nNo calls have completed.’,flush= True) if i == 0 else print(f’done.\n{i} calls have completed’,flush=True)
            i = 0
        if test[0]==’prediction’:
            for call in self.active_calls_test:
                if call[3] < current_time:
                    call[0].isBusy = False #free the caller
                    call[1].isBusy = False #free the callee
                    self.RATs_test[call[2]-1]+=Simulator.voice_RB
                    self.active_calls_test.remove(call)

        #get user group
        grp = group_rng.choice([’A’,’B’],p=[groupA,1-groupA])

```

```
        #get the new caller(could be the same previous caller)
        caller = user_rgn.choice(self.predictor.dataGenerator.users)

        #get the caller that is not busy
        trials = np.random.randint(20,50) #number of times to try getting the call
        while caller.isBusy and trials > 0:
            trials-=1
            caller = np.random.choice(self.predictor.dataGenerator.users)
        if trials<= 0: # if couldn’t find the call in given trials exit
            continue
        #get current time if 24-hour format
        time = current_time.total_seconds()/3600.0

        #now place the call
        call = caller.dial_call_v1(time,generator=caller_gen)

        #check if the call was successful
        if call is None:
            continue
        callee, dur,avg = call

        #limit the duration to within the boundaries appropriate fo the test been run
        if test[0] == ’duration’ and test[1] == ’short’:
            #dur = min(dur, mean-std)
            while dur > mean-std:

                callee, dur,avg= caller.dial_call_v1(time,generator=caller_gen)
        elif test[0]== ’duration’ and test[1] == ’medium’:
            #dur = np.clip(dur,mean-std,mean+std)
            while dur < mean-std or dur > mean+std:

                callee, dur,avg= caller.dial_call_v1(time,generator=caller_gen)

        elif test[0]== ’duration’:
            #dur = max(dur,mean+std)

            while dur < mean+std:
                #resample users again, still duration condition is meet
                caller = user_rgn.choice(self.predictor.dataGenerator.users)
                callee, dur,avg= caller.dial_call_v1(time,generator=caller_gen)


        #now admit the call
        rel = self.predictor.dataGenerator.R[caller.ID-1][callee.ID-1]

        #predictor duration
        pred_dur = self.predictor.predict([rel,User.adjust_time(time),avg])

        #get visibility time at random
```

```
        vis = vis_rng.beta(2,5)*770

        if test[0] == 'visibility' and test[1] == 'short':
            vis = min(vis,200)

        elif test[0] == 'visibility' and test[1] == 'medium':
            vis = np.clip (vis, 201,400)

        elif test[0] == 'visibility':
            vis = max(vis,400)


        #admit the call
        RAT = self.admission_control(pred_dur,vis,grp,dur)

        if test[0]== 'prediction':
            RAT_test = self.admit_no_duration(pred_dur,vis,grp)

            if RAT_test >0:
                self.active_calls_test.append((caller,callee,RAT_test,current_time+timedelta(seconds
                    =dur)))

        #add to active calls ro proceed
        active_calls.append((caller,callee,RAT,current_time+timedelta(seconds=dur))) if RAT > 0 else
             active_calls
        if verbose:
            exec(status,globals(),locals())

        if verbose and test[0] == 'prediction':
            exec(status_test,globals(),locals())


    return None
Simulator.simulate = simulate
```

Simulation Results

```


def effect_of_duration_prediction(start=12,time=5.0/18.0,groupA=1.0,verbose=False):
    """
    Simulate the network in a scenario when service duration is considered, and when it is not

    The functions simulates the network to investigate the effect on considering service duration
    on network performance. It gathers data and returns it

    Parameters:
    ----------
```

```python
    start: float
        the time of the day when to start the simulation ( if important for getting call start time for predictive model)
    time: float
        total simulation time in hours.
    grp: float
        probability of group A users in the network
    verbose:
        print the progress results during simulation

    Returns:
        the simulation data of the experiment
    """
    CBP = []
    SHP = []
    CBP_test = []
    SHP_test = []
    sim = Simulator()
    if verbose:
        print('************Simulation Progress*****************',flush=True)

    for rate in Simulator.rates:
        if verbose:
            print(f'\nsimulation with arrival rate set to {rate} calls/sec...',flush=True,end='')
        sim.simulate(start,time,arrival_rate=rate,test=['prediction'],groupA=groupA)
        if verbose:
            print('done.\nCollecting data...',end='',flush=True)
        CBP.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
        CBP_test.append(sum(sim.blocked_test)/sum(sim.attempts_test) if sum(sim.attempts_test)!=0 else 0)
        SHP.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
        SHP_test.append(sum(sim.handoffs_test)/sum(sim.rat_1_calls_test) if sum(sim.rat_1_calls_test)!=0 else 0)
        if verbose:
            print(f'done.\n')
            print(f'*************Results Summary***************',flush=True)
            print(f'Total Call Attempts:\n\tWith No Duration Prediction:    {sum(sim.attempts_test)}',flush=True)
            print(f'\tWith Duration Prediction: {sum(sim.attempts)}')
            print(f'Total Blocked:\n\tWith No Duration Prediction:{sum(sim.blocked_test)}',flush=True)
            print(f'\tWith Duration Prediction:{sum(sim.blocked)}')
            print(f'Total Satellite handoffs:\n\tWith No Duration Prediction: {sum(sim.handoffs_test)}',flush=True)
            print(f'\tWith Duration Prediction: {sum(sim.handoffs)}',flush=True)
            print(f'Total Users in RAT-1:\n\tTotal with No Duration Prediction: {sum(sim.rat_1_calls_test)}',flush=True)
            print(f'\tTotal with Duration Prediction: {sum(sim.rat_1_calls)}',flush=True)
    return CBP,SHP,CBP_test,SHP_test
```

```

#Run the function and collect data
#set verbose to True to see the  simulation progress
#change groupA to vary the proportion of group A users. 1.0 means 100% users are from group A


pred_data = effect_of_duration_prediction(start_hour,sim_time,groupA=1,verbose = True)

#plot call blocking probabilities

duration_CBP = pred_data[0] #call blocking probabilities when duration is considered
no_duration_CBP = pred_data[2] #call blocking probabilities when no duration is considered

#plot the arrival rates VS call blocking probabilities
plt.plot(Simulator.rates, no_duration_CBP,label='no duration considered',ms=6,marker='^',ls='solid')
plt.plot(Simulator.rates, duration_CBP,label='duration considered',ms=3,marker='o',ls='dashed')

plt.grid(True)
plt.xticks(Simulator.rates)
plt.xlabel('service arrival rates')
plt.ylabel('call blocking probabilities')
plt.title('call blocking probability when duration is considered and when it is not')
plt.legend()
plt.show();

duration_SHP = pred_data[1] #satellite handoff probabilities when duration is considered
no_duration_SHP = pred_data[3] #satellite handoff probabilities when no duration is considered

plt.plot(Simulator.rates, no_duration_SHP,label='no duration considered',ms=5,marker='^',ls='solid')
plt.plot(Simulator.rates, duration_SHP,label='duration considered',ms=5,marker='o',ls='dashed')

plt.grid(True)
plt.xticks(Simulator.rates)
plt.xlabel('service arrival rates')
plt.ylabel('satellite handoff probabilities')
plt.title('call blocking probability when duration is considered and when it is not')
plt.legend()
plt.show();

def effect_of_duration(start=12,time=5.0/18.0,verbose=False,grp=1):
    """
    Simulates the network to investigate effect of duration on network perfomance.
    """
    data =  {} #store simulation data for each scenario
    scenarios = ['short','medium','long'] #duration scenarios
    if verbose:
        print('************Simulation Progress****************',flush=True)

    sim = Simulator()
```

```python

    for scenario in scenarios:
        if  verbose:
            print(f'\nSimulating scenario where durations are {scenario}:',flush = True)
        CBP = []
        SHP =  []

        for rate in Simulator.rates:
            if verbose:
                print(f'\tSimulating with arrival rate set to {rate} calls/sec...',end= '',flush =
                    True)

            sim.simulate(start,time,arrival_rate = rate,test = ['duration',scenario])
            CBP.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
            SHP.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
            if verbose:
                print('done.\n\t*********Results Summary**********',flush= True)
                print(f'\tTotal Call Attempts:{sum(sim.attempts)}',flush=True)
                print(f'\tTotal Blocked Calls:{sum(sim.blocked)}',flush=True)
                print(f'\tTotal Satellite Handoffs:{sum(sim.handoffs)}',flush=True)
        data[scenario] = (CBP,SHP)
    return data
#Run the function to generate data to plot. It takes thr followin parameters:
#
#start_hour and start_time, as specified in the parameters sections
#grp must be a value between 0 and 1 indicating the percentage of group A users in the network
#verbose set this to True to see the simulation progress
dur_data = effect_of_duration(start_hour,sim_time,grp=1.0,verbose=True)

#plot call blocking pobabilities


plt.plot(Simulator.rates, dur_data['short'][0],label='short durations',marker='^',ms=5)
plt.plot(Simulator.rates,dur_data['medium'][0],label='medium durations',marker='s',ms=5,ls='dashed')
plt.plot(Simulator.rates,dur_data['long'][0],label='long durations',marker='o',ms=5,ls='dotted')

plt.xlabel('service arrival rates')
plt.ylabel('call blocking probabilities')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.title('call blocking probabilities for different duration scenarios')
plt.legend()
plt.show()


# plot satellite handoff probabilities



```

```python
plt.plot(Simulator.rates, dur_data['short'][1],label='short durations',marker='^',ms=5)
plt.plot(Simulator.rates,dur_data['medium'][1],label='medium durations',marker='s',ms=5,ls='dashed')
plt.plot(Simulator.rates,dur_data['long'][1],label='long durations',marker='o',ms=5,ls='dotted')

plt.xlabel('service arrival rates')
plt.ylabel('satellite handoff probabilities')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.title('satellite handoff probabilities for different duration scenarios')
plt.legend()
plt.show

def effect_of_visibility_time(start=12,time=5.0/18.0,verbose=False,grp=1):
    """
    simulates the network and collects simulation data to investigate the effect
    of visibility time of network performance.

    Parameters:
    ----------
    start: float
        the time of the day when to start the simulation ( if important for getting call start time
            for predictive model)
    time: float
        total simulation time in hours.
    grp: float
        probability of group A users in the network
    verbose:
        print the progress results during simulation

    Returns:
        the simulation data of the experiment
    """
    data = {}
    scenarios = ['short','medium','long']
    sim = Simulator()
    if verbose:
        print('************Simulation Progress****************',flush=True)
    for scenario in scenarios:
        if verbose:
            print(f'\nSimulating scenario for {scenario} visibility time:',flush=True)
        cbp = []
        shp = []
        for rate in Simulator.rates:
            if verbose:
                print(f'\tsimulating with arrival rate set to {rate} calls/sec...',flush = True,end=
                    '')
            sim.simulate(start,time,arrival_rate=rate,test=['visibility',scenario],groupA=grp)
            cbp.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
            shp.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
```

```
            if verbose:
                print('done.\n\t*********Results Summary**********',flush= True)
                print(f'\tTotal Call Attempts:{sum(sim.attempts)}',flush=True)
                print(f'\tTotal Blocked Calls:{sum(sim.blocked)}',flush=True)
                print(f'\tTotal Satellite Handoffs:{sum(sim.handoffs)}',flush=True)

        data[scenario] = (cbp,shp)
    return data

#secify the following parameters:
#start_hour and start_time, as specified in the parameters sections above
#grp must be a value between 0 and 1 indicating the percentage of group A users in the network
#verbose set this to True to see the simulation progress
vis_data =effect_of_visibility_time(start_hour,sim_time,verbose=True,grp=1)

#plot call blocking probabilities

plt.plot(Simulator.rates,vis_data['short'][0],label='short visibilities',marker='^',ms=9)
plt.plot(Simulator.rates,vis_data['medium'][0],label='medium visibilities',marker='s',ms=6,ls='
    dashed')
plt.plot(Simulator.rates,vis_data['long'][0],label='long visibilities',marker='o',ms=3,ls='dotted')

plt.xlabel('service arrival rates')
plt.ylabel('call blocking probabilities')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.title('call blocking probabilities for different visibility scenarios')
plt.legend()
plt.show()

# Plot satellite handoff probabilities

plt.plot(Simulator.rates,vis_data['short'][1],label='short visibilities',marker='^',ms=9)
plt.plot(Simulator.rates,vis_data['medium'][1],label='medium visibilities',marker='s',ms=6,ls='
    dashed')
plt.plot(Simulator.rates,vis_data['long'][1],label='long visibilities',marker='o',ms=3,ls='dotted')

plt.xlabel('service arrival rates')
plt.ylabel('satellite visibilities probabilities')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.title('satellite handoff probabilities for different visibility scenarios')
plt.legend()
plt.show()

def effect_of_user_group(start=12,time=5.0/18.0,verbose=False):
    """
    simulates the network with different groups of users and collects data for each case.

```

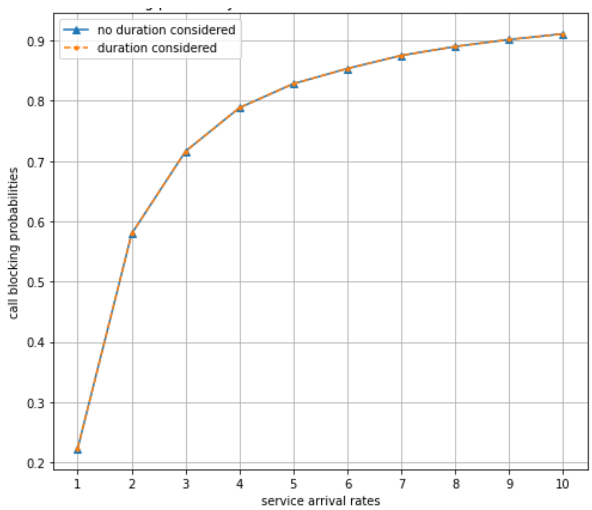
```
    start: float
        the time of the day when to start the simulation ( if important for getting call start time
            for predictive model)

    time: float
        total simulation time in hours.

    verbose:
        print the progress results during simulation

    Returns:
        the simulation data of the experiment

    """
    groups = {'100%-A':1,'80%-A':0.8,'20%-A':0.2,'0%-A':0}
    data = {}
    sim = Simulator()
    if verbose:
        print('************Simulation Progress****************',flush=True)
    for g in list(groups.keys()):
        if verbose:
            print(f'\nSimulating the scenario when there are {groups[g]*100}% of group A users:',
                flush = True)
        cbp = []
        shp = []
        for rate in Simulator.rates:
            if verbose:
                print(f'\tsimulation with arrival rate set to {rate} calls/sec...',flush = True,end=
                    '')
            sim.simulate(start,time,arrival_rate=rate,groupA=groups[g])
            if verbose:
                print('done.\n')
            cbp.append(sum(sim.blocked)/sum(sim.attempts) if sum(sim.attempts)!=0 else 0)
            shp.append(sum(sim.handoffs)/sum(sim.rat_1_calls) if sum(sim.rat_1_calls)!=0 else 0)
            if verbose:
                print('\t***********Results Summary**********',flush=True)
                print(f'\tTotal Call Attempts:\n\t\tGroup A calls:{sim.attempts[0]}\n\t\tGroup B
                    calls:{sim.attempts[1]}',flush=True)
                print(f'\tTotal Blocked Calls:\n\t\tGroup A blocks:{sim.blocked[0]}\n\t\tGroup B
                    blocks:{sim.blocked[1]}',flush=True)
                print(f'\tTotal Satellite Handoffs:\n\t\tGroup A handoffs:{sim.handoffs[0]}\n\t\
                    tGroup B handoffs:{sim.handoffs[1]}',flush=True)

        data[g] = (cbp,shp)
    return data
#set the verbose to True to see simulation progress

grp_data = effect_of_user_group(start_hour,sim_time,verbose=True)
```

```
#plot call blocking probability of individual user groups

plt.plot(Simulator.rates,grp_data['100%-A'][0],label='100% group A',marker='s',ls='dashed',ms=5)
plt.plot(Simulator.rates,grp_data['0%-A'][0],label='100% group B',marker='s',ls='dotted',ms=5)
plt.xlabel('service arrival rates')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.ylabel('call blocking probabilities')
plt.legend()
plt.title('call blocking probabilities when there 100% of each group')
plt.show()

# plot satellite handoff probabilities of user groups

plt.plot(Simulator.rates,grp_data['100%-A'][1],label='100% group A',marker='s',ls='dashed',ms=5)
plt.plot(Simulator.rates,grp_data['0%-A'][1],label='100% group B',marker='^',ls='dotted',ms=5)
plt.xlabel('service arrival rates')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.ylabel('satellite handoff probabilities')
plt.legend()
plt.title('satellite handoff probabilities when there 100% of each group')
plt.show()

plt.plot(Simulator.rates,grp_data['100%-A'][0],label='100% group A',marker='s',ls='solid',ms=8)
plt.plot(Simulator.rates,grp_data['80%-A'][0],label='80% group A',marker='o',ls='dashed',ms=5)
plt.plot(Simulator.rates,grp_data['20%-A'][0],label='20% group A',marker='^',ls='dotted',ms=3)
plt.plot(Simulator.rates,grp_data['0%-A'][0],label='0% group A',marker='.',ls='dashdot',ms=5)

plt.xlabel('service arrival rates')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.ylabel('call blocking probabilities')
plt.legend()
plt.title('call blocking probabilities when different proportions of each group')
plt.show()

plt.plot(Simulator.rates,grp_data['100%-A'][1],label='100% group A',marker='s',ls='solid',ms=5)
plt.plot(Simulator.rates,grp_data['80%-A'][1],label='80% group A',marker='s',ls='dashed',ms=5)
plt.plot(Simulator.rates,grp_data['20%-A'][1],label='20% group A',marker='s',ls='dotted',ms=5)
plt.plot(Simulator.rates,grp_data['0%-A'][1],label='0% group A',marker='s',ls='dashdot',ms=5)

plt.xlabel('service arrival rates')
plt.xticks(Simulator.rates)
plt.grid(True)
plt.ylabel('satellite handoffs')
plt.legend()
plt.title('satellite handoff probabilities when different proportions of groups')
plt.show()
```

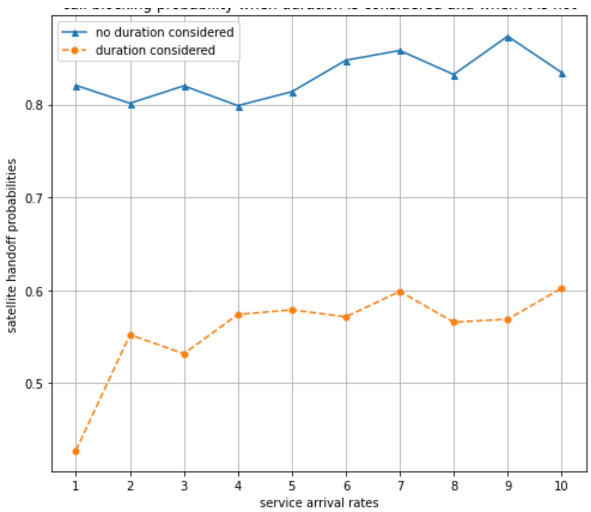
Chapter 9

Appendix C

C: Additional Simulation Results

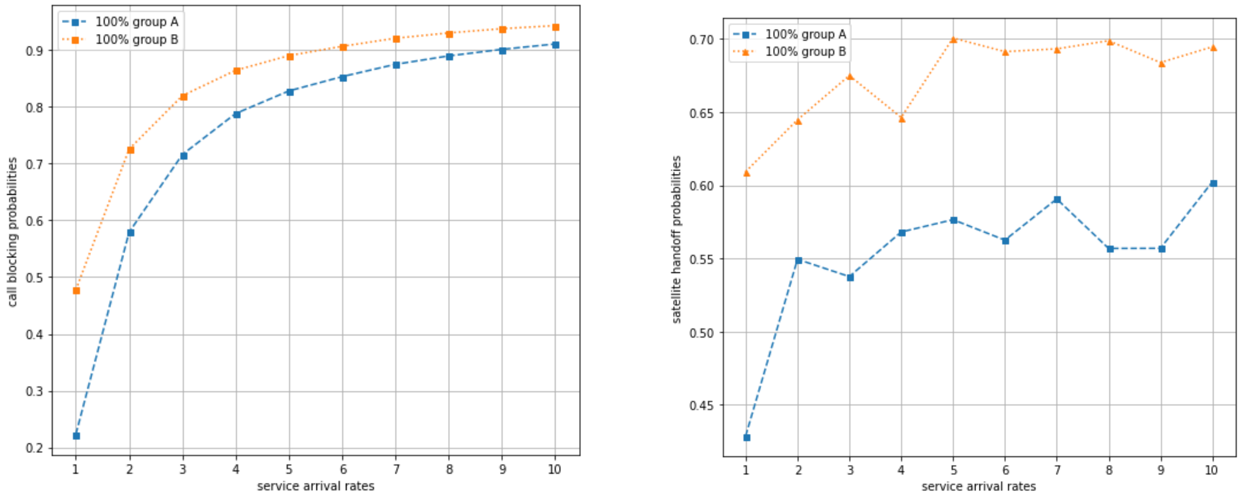


(a) Effect of admission with and without duration prediction on call blocking probability



(b) effect of admission with and without duration prediction on satellite handoff probability

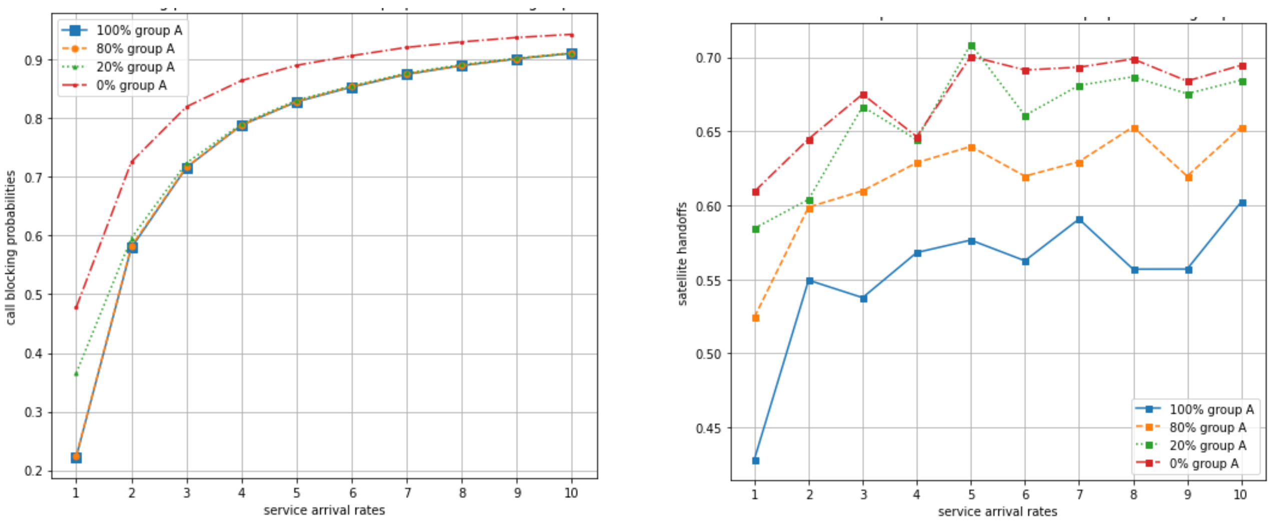
Effect of admission with and without duration prediction on network performance



(a) Effect of individual user group on call blocking probability

(b) effect of individual user group on satellite handoff probability

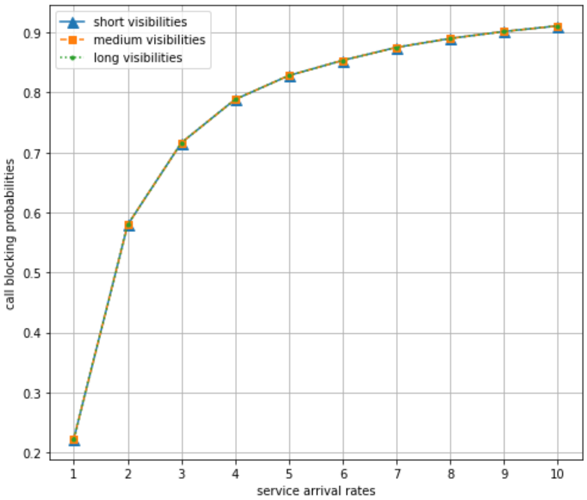
Effect of individual user group on network performance



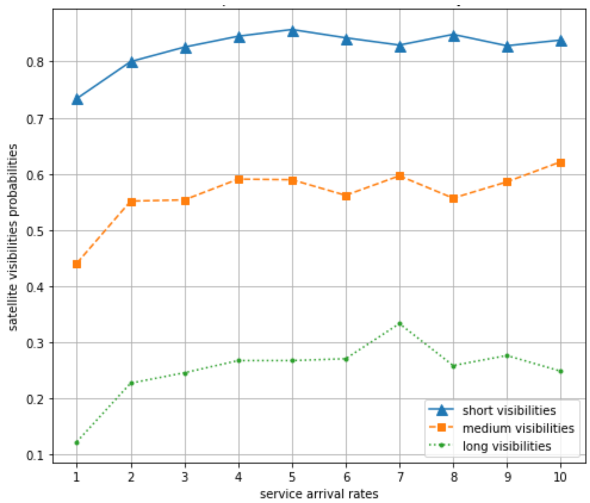
(a) Effect of mixed proportions of user groups on call blocking probability

(b) Effect of different proportions of user groups on satellite handoff probability

Effect of individual user group on network performance

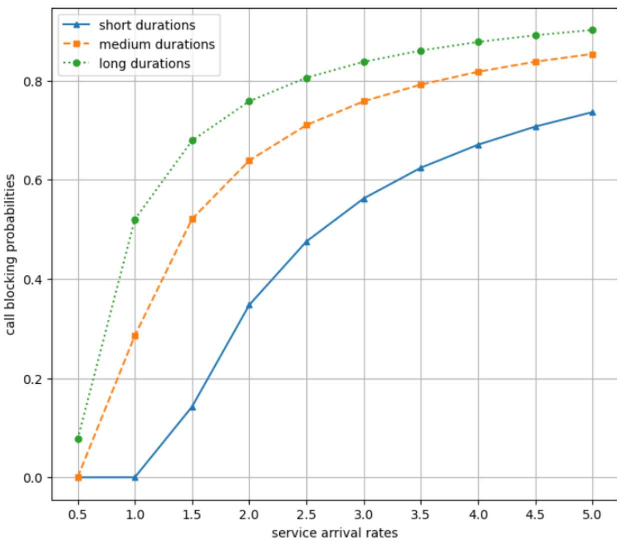


(a) Effect of satellite visibility time on call blocking probability

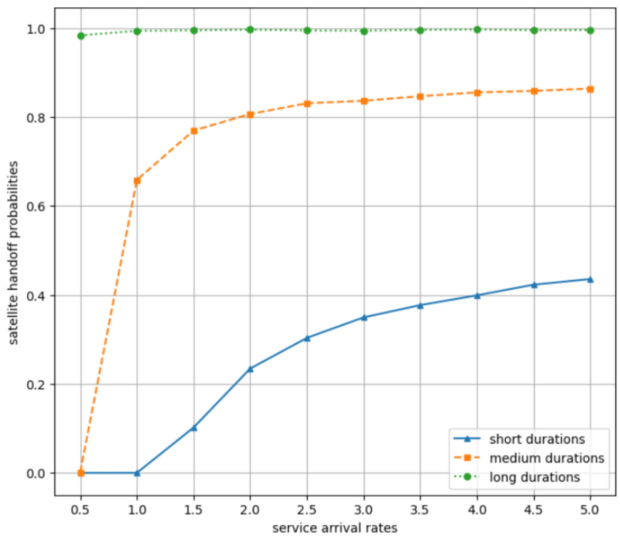


(b) Effect of satellite visibility time on satellite handoff probability

Effect of visibility time on network performance



(a) Effect of different service duration scenarios on call blocking probability



(b) Effect of different service duration scenarios on satellite handoff probability

Effect of service duration on network performance