SEAMLESS HANDOVER FOR PERVASIVE SPEECH COMMUNICATION

By

Vijaya Nirmala Mitnala

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

September 2024

The intended audience for this thesis comprises scholars, researchers, and professionals in the fields of network engineering, audio signal processing, and machine learning. It is designed to offer a comprehensive exploration of **Seamless Handover for Pervasive Speech Communication** and is particularly relevant to those with a vested interest in device detection based on multivariate audio features and application layer session handovers. Additionally, this work may be of interest to educators and students seeking a deeper understanding of applying machine learning techniques for processing audio data. The thesis thoroughly examines and analyses session handling and audio features using machine learning techniques, providing a valuable resource for those aiming to expand their knowledge in this area and for practitioners seeking insights to inform their work in pervasive speech.

Key words– Pervasive speech system, Vertical network handover, Horizontal device handover, Audio signal processing, Deep learning, Session Initiation Protocol.

Acknowledgements

I would like to express my deepest gratitude and appreciation to all those who have contributed to the completion of this thesis.

I would first and foremost like to thank my supervisor, Professor Martin J. Reed, for his guidance, support, and expertise throughout research process. His constant feedback, valuable suggestions, and continuous encouragement have been instrumental in shaping the direction and quality of this work.

I express my heartfelt thanks to the BT (British Telecommunications Plc.) research delegates, Ian Kegel and John Bicknell, for their valuable suggestions and support. I also want to convey my deep appreciation to my manager, Jamie Mcquillan, from BT, for providing invaluable guidance and unwavering support throughout the completion of this thesis.

Furthermore, I extend my sincere gratitude to my friends and colleagues who have supported me in various ways, offering invaluable insights, feedback, engaging in stimulating discussions, and creating a positive and motivating environment.

Finally, I would like to convey my profound appreciation to my family, particularly my dear brother, Rama Krishna Rao Mitnala. Their unconditional love, encouragement, and understanding have been my constant source of strength throughout this journey. I am forever grateful for their belief in me and their constant support.

Declaration

I hereby declare that this thesis represents my original work and has not been previously submitted for any other degree or diploma at any university or institution of higher education.

I confirm that, to the best of my knowledge, the intellectual content contained within this thesis is solely my own creation, and all the assistance received in preparing this thesis and sources have been acknowledged.

Vijaya Nirmala Mitnala

September 2024

Abstract

The consistent growth of smartphones and the smart speaker market has played a vital role in establishing high-quality, far-field speech communication as a feasible alternative to traditional handsets. The prevalence of smartphones, multiple smart speakers and various computing platforms in many homes has led to much greater flexibility in how and where users communicate. This trend is expected to persist as next-generation voice and media services, such as Augmented Reality (AR)/Virtual Reality (VR), become more common. Additionally, key services such as financial and healthcare, for which flexible, high-quality communications are crucial components, are transitioning online. Despite these trends, comparatively little has been done to offer a converged communications experience once a speech call session is in progress. For example, the simple act of switching an ongoing call from a smartphone to a smart speaker is often a highly manual process. Pervasive communication systems aim to address this by providing a seamless, flexible communication experience across multiple devices and, where required, multiple networks. The primary contributions of this thesis provide solutions for both vertical handovers (transitions between heterogeneous networks) and horizontal handovers (transitions within a homogeneous network domain, specifically involving smart devices connected to the same network). This work uses a supervised machine learning based approach to predict user's transitions between the networks, and thus, overcome interruptions in speech due to signalling handover. For device handovers, this work proposes processing multivariate signalling features with time-series prediction algorithms and the deep learning techniques to accurately determine the most suitable device for the user for the handover. Additionally, this thesis considers how the Session Initiation Protocol (SIP) can be used in IP telephony systems where a seamless transition is required in a scope of handover between the networks and between the devices, while also proposing new solutions to achieve seamless session handovers.

Contents

List of Figures

List of Tables

Acronyms

1DCNN One-Dimensional Convolutional Neural Networks

3GPP 3rd Generation Partnership Project

A Signal Magnitude

AR Augmented Reality

B2BUA Back-to-Back User Agent

CNNs Convolutional Neural Networks

DCT Discrete Cosine Transform

FFT Fast Fourier Transform

IETF Internet Engineering Task Force

IMS IP Multimedia Subsystem

LTE Long Term Evolution

LSTM Long Short-Term Memory

ML Machine Learning

MSC Magnitude Squared Coherence

MFCC Mel Frequency Cepstral Coefficients

RAN Radio Access Network

SIP Session Initiation Protocol

SDP Session Description Protocol

SSL Sound Source Localisation

UAC User Agent Client

UAS User Agent Server

VR Virtual Reality

VOIP Voice Over IP

Wi-Fi Wireless Fidelity

List of Publications

Conference Publications

C1. **V. N. Mitnala**, M. J. Reed, I. Kegel and J. Bicknell, "Avoiding handover interruptions in pervasive communication applications through machine learning," 2021 IEEE International Conference and Expo on Real Time Communications at IIT (RTC), 2021, pp. 1-8, doi: 10.1109/RTC52972.2021.9615673 (labeled as C1.)

C2. **V. N. Mitnala**, M. J. Reed, I. Kegel and J. Bicknell, "Seamless device handover for pervasive speech communication," 2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA), Cairo, Egypt, 2022, pp. 1-6, doi: 10.1109/ICCSPA55860.2022.10018978 (labeled as C2.)

Journal Publications

J1. **V. N. Mitnala**, M. J. Reed, J. Bicknell and JoyRaj Chakraborty, "Intelligent speech handover for smart speakers through deep learning: a custom loss function approach". (labeled as J1.) - Submitted to IEEE Transactions on Consumer Electronics

1

Introduction to Thesis

In this chapter, an exploration is undertaken regarding the pervasive speech system, problem statement, its motivation, and the objectives underpinning the work presented in this thesis. This chapter also delves into the principal contributions and the overview of the work presented in this thesis. Subsequently, a comprehensive breakdown is provided regarding the organization of the ensuing sections of this thesis.

1.1 Problem Statement

Given the continuous advancement of the Internet and heterogeneous communication networks, along with the growing trend of utilising diverse devices, the way we access speech communication provided by technology and devices is about to shift to a pervasive audio view. This future trend, and associated technical challenges, are predicted by industry but with less coverage in academic venues. Thus, this first chapter will draw upon industry statements before moving to the underpinning academic research in Chapter 2.

The prevalence of smartphones, multiple smart speakers, and various computing platforms in many households has introduced greater flexibility in how and where users engage in communication. This trend is expected to persist, especially with the growing prevalence of the next generation of voice and media services [1]–[3], AR/VR [1], and the migration of several services like healthcare to online platforms, emphasizing the need for flexible, high-quality communications. Despite these evolving patterns, there has been relatively little effort to provide a unified communications experience once a speech call session is under progress. There is session handover latency when there is a network switch, resulting in speech interruptions also the process of switching an ongoing call from a smartphone to a smart speaker is often a manual and cumbersome task. This opens up the research scope to seek solutions for seamless handovers to achieve a pervasive speech experience.

1.2 Motivation

This thesis introduces the term *pervasive speech communication* which aims to deliver a seamless audio and/or video experience based on situational context – such as the preferred end device or best available network – both before and during a call session. It is said to be *pervasive* as it moves away from a single dedicated end-terminal and instead,

[1] Augmented Reality (AR) is an interactive experience that overlays digital content onto the real world, enhancing the user's perception through visual, auditory, and other sensory elements. Virtual Reality (VR) creates a completely immersive environment that replaces the real world with a simulated one, allowing users to interact solely within a computer-generated space.

an appropriate end-terminal is automatically selected for the user to enable a smooth user experience. In order to achieve this, pervasive speech systems need to facilitate seamless transitions between various access and backhaul networks as needed (such as 5G and 4G Radio Access Network (RAN), and private or public Wireless Fidelity (Wi-Fi) with broadband backhaul), a process known as *vertical handover*. Additionally, these systems must support different audio presentation systems (such as one or more smart speakers, smartphones, tablets, and PCs) within the same network, referred to as *horizontal handover*. Despite the tremendous growth of IP telephony in recent years, and the diversity of devices and networks on which it is now used [4]–[6], support for pervasive communications remains limited. Mobile network standards [7] acknowledge the significance of IP mobility management mechanisms in facilitating a seamless transition between network types. However, these standards do not address transitions between devices, and also there are still instances of handover interruptions due to session transition latency. Smart speaker providers have added new features to their respective smart speaker offerings [8]–[11] to share communication sessions across multiple devices. However, they still do not take all the advantages of having multiple devices to improve the user communication experience. There exists an opportunity to substantially enhance speech communication across multiple devices by:

- Making decisions to transfer communication speech between devices independently of conscious user behavior *i.e.* to be able to switch devices independent of user interaction and not simply switch at an arbitrary fixed proximity to a device
- Using a variety of measurements associated with each smart device (not necessarily made on the device) in order to assess its suitability for providing communication at any one point in time
- Optimising the user experience by allowing more than one device to be used simultaneously to account for differing acoustic capabilities

This thesis aims to tackle and offer solutions to the aforementioned scope of speech communication across multiple devices. The focus of this work is to provide a solution

for the seamless speech transition between the devices within the same network domain, *i.e.* addressing horizontal handovers between devices. Additionally, considering the inclusive scope of the pervasive speech system, which extends to handovers between different networks, referred to as vertical handovers, this thesis also puts forward solutions for seamless vertical handovers. This will be achieved by proposing effective solutions through the application of advanced techniques. The ultimate goal is to provide a pervasive communication experience by eliminating handover interruptions and avoiding the need for manual call transfers across multiple devices. Figure 1.1 illustrates the scope of this thesis's pervasive speech system, with prominent use cases including:

1. Seamless session handover during vertical network transition
    (a) For example: LTE to Wi-Fi network transition
2. Device detection and seamless session handover during horizontal device transition
    (a) For example: Smartphone to Smart speaker

This study will concentrate on offering solutions for the mentioned use-cases. The research considers SIP for application layer session handling. SIP is a heavily used and widely accepted application layer protocol in Voice Over IP (VOIP) and IP Multimedia Subsystem (IMS) systems. Chapter 2 provides an in-depth examination of SIP. This study extensively explores SIP's essential mobility features concerning session management and media transmission across networks and devices. Various methods can accomplish these multiple transmissions, such as repeated unicast transmissions, layer 3 multicast, or layer 4 multipath. Section 2.2.2 explores these alternatives in detail. However, this thesis concentrates exclusively on multiple unicast transmissions, providing modified SIP solutions for seamless media delivery, as other techniques are not widely supported by current networks. The research capitalizes on all available SIP features and suggests strategies to reduce or prevent session handovers with minimal alterations to the existing SIP protocol. The author is already an expert on SIP operations, having worked on the SIP stack for BT (British Telecommunications Plc.). However, the extensions presented in this thesis are beyond the existing SIP standards and systems.

Additionally, this research explores machine learning techniques, both supervised and deep learning, along with the domain of audio signal processing. The first use-case involves employing machine learning techniques on SIP signalling, while the second use-case utilises deep learning and time-series predictors on audio signal features. Furthermore, SIP signalling is employed for session handover across devices in this second use case. This pervasive speech system does not exist so far and thus outcome of this thesis meets the upcoming user requirements.

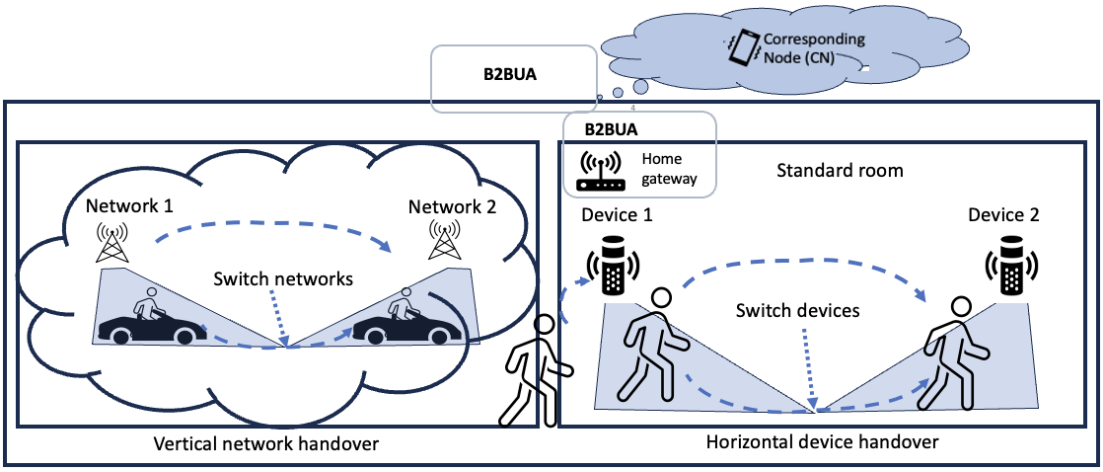

Figure 1.1: Pervasive speech system

1.3 Research Questions

Can SIP-based speech communication have seamless handover between networks?

- Can we avoid the SIP call establishment time of >=280 ms?
- Can we do a change in the attachment through just SIP application layer processing?

Can SIP-based speech communication have seamless handover between acoustically far-field terminals (e.g. smart speakers) in the same network?

- Can media interruptions during SIP session handover be avoided?
- Can the appropriate smart speakers be detected from speech signals alone without knowing the device locations?

1.4 Contribution

The following subsections summarise the primary contributions, addressing the research questions outlined in Section 1.3. They highlight the novel insights, objectives, and advancements presented in this thesis, which aim to achieve the two use cases of the pervasive speech system. In Figure 1.1, the left side *i.e.* vertical network handover covers the use-case 1, and the right side *i.e.* horizontal device handover covers the use-case 2. In addressing the requirements of these use-cases, this study answers the mentioned research challenges.

1.4.1 Use-case 1: seamless vertical handover

To fulfill the requirements of use-case 1, this thesis proposes a proactive network handover solution for optimising session handling during network handovers. A supervised machine learning approach is presented to predict network mobility, allowing media delivery to be established before the actual physical transition, referred to as "make before move". As depicted in Figure 1.2, during a call transition from network N1 to Ni with the physical transition occurring at network attachment point p, this study suggests a strategy of forecasting one or two most likely future network attachment points (*p1*, *p2*) and transmitting the media to p, *p2*, and *p3*. Subsequently, the speech is conveyed to the anticipated network endpoints prior to the physical move, a technique referred to as *Bicasting*/*TriCasting*. This proactive speech transmission facilitates a *make-before-break* handover, effectively preventing speech interruptions throughout the transition. The proposed solution is designed for application layer session handling using SIP, making use of SIP's *terminal mobility* feature and SIP supported extension headers. This proposed solution is evaluated using a real-world vehicular mobility dataset. Further details of this proposed solution is available in Chapter 3. This work has been published in a conference paper at IEEE International Conference and Expo on Real Time Communications at IIT (RTC), 2021 [C1].

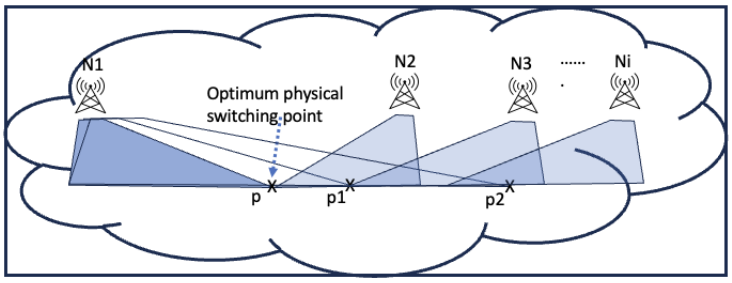

Figure 1.2: Proactive media handover to the future network attachment points p1 and p2 alongside with p

1.4.2 Use-case 2: seamless horizontal handover

Use-case 2 necessitates a smooth call transfer during horizontal handover, involving the transition between devices within the same network domain, such as within a confined space like a room, referring to the right side of Figure 1.1. To fulfil this requirement, two key steps are identified: firstly, the automatic identification of a suitable device, and secondly, the seamless transfer of the ongoing call to the detected device.

This research drew inspiration from a proof of concept conducted at the BT lab, where speech signal features like the direction of arrival (DOA) were used to identify the device. However, upon validating the results from the lab experiment, it was evident that the proposed solution does not effectively address real-world scenarios characterized by locus user movements, room reflections, and uncalibrated systems. In response, the initial research for this thesis conducted a literature review, as described in Chapter 2, to understand methodologies for audio location techniques. This revealed that the Magnitude Squared Coherence (MSC) is the most promising technique used in the literature for determining the location from audio signals alone. The MSC will be fully described in Chapter 2, however, it can be briefly described as a method to determine how much an audio source has been influenced by the audio environment. and thus the solution leverages the MSC signal data captured on the devices to identify the suitable device. The talker's voice serves as the signal source, and MSC is computed from all scoped devices at each talker's position, even while in motion. No calibration is required for this solution, unlike other systems which rely upon external measurements and calibration [12]. The difference in MSC values across devices is calculated and the device

with higher MSC will be the most suitable device to continue the conversation. To assess the viability of this proposed solution, simulations were conducted. The experimental results revealed significant variations in raw speech signal features due to environmental factors, particularly room effects. To address this challenge, the smoothing out of MSC values using a double exponential smoothing predictor was introduced, resulting in improved performance. To facilitate session handover to the identified device, this study suggests parallel sessions and proposes alternatives to SIP signalling. It explores the utilisation of the SIP *personal mobility* feature to achieve a smooth transition of media. Additional information about this suggested approach can be found in Chapter 4. This work has been published in a conference paper in 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA), Cairo, Egypt, 2022 [C2].

Additional exploration has been conducted to address real-world scenarios in this use-case. This has resulted in the formulation of a robust solution that addresses issues not covered by the above base solution. Deep learning techniques are employed on speech signalling features for device detection. Various audio features are analyzed for potential use in device detection. Given the non-linearity of these audio features, One-Dimensional Convolutional Neural Networks (1DCNN) are deemed suitable as the deep learning technique of choice. The required training data is collected by simulating a room layout furnished with multiple smart devices with microphone arrays. Simulated sound sources are introduced, and various reverberation times are incorporated to emulate authentic sound environments. A thorough assessment involving different datasets with various audio signals, movement patterns, and different room scenarios establishes the efficacy of the proposed approach in accurately predicting the most suitable device. Further details regarding this proposed approach are available in Chapter 5. This work has been submitted to the IEEE Transactions on Consumer Electronics, 2024 [J1].

1.4.3 Summary of research contributions

The research contributions described above over the two use cases can be summarised as:

- A modified SIP signalling is proposed for seamless media transmission during both vertical and horizontal session handovers. These SIP modifications require only local i.e. home gateway level changes, rather than network-wide changes (This work is covered in Chapters 3 and 4).
- Seamless vertical session handover (Use-case 1) between the networks using supervised machine learning (This work is covered in Chapter 3).
- Seamless horizontal session handover in far-field (Use-case 2) using:
  - ◇ MSC to predict the best smart device (This work is covered in Chapter 4).
  - ◇ A 1DCNN deep learning technique to improve the MSC based predictor (This work is covered in Chapter 5).

1.5 Research Methodology

This research adopts a data-driven approach, an evidence-based methodology that relies on data analysis and interpretation to inform decision-making processes and strategies. The methodology is detailed by the authors, Maass et al. in their work [13]. Unlike theory-driven approaches, it is particularly useful for machine learning techniques to extract meaningful patterns, trends, and insights from data sets. This study utilises real-world mobility datasets, acoustic datasets, and data generated through acoustic room simulation. In the absence of a pervasive speech system, a realistic vehicular mobility dataset from the city of Cologne in Germany [14], [15] is employed to validate the vertical handover solution, with detailed information available in Chapter 3. For the validation of the device handover solution, an acoustic dataset is generated by simulating the acoustic room using Pyroomacoustics [16], an open-source Python library designed for simulating room acoustics and sound propagation. The talker's movements are simulated using a

cubic Bezier curve, and various audio signals from [17], replicating the talker's voice, are examined. Additionally, the solution is verified using a real room/talker dataset obtained from [18], [19]. The audio signal processing and feature extraction are conducted using the Scipy Python libraries [20] and Librosa Python libraries [21]. Further insights into the acoustic room simulation and acoustic data extraction are presented in Chapters 4 and 5. Subsequently, established machine learning algorithms from SciKit Learn [22] and Tensorflow Keras [23] are employed to predict network handover and device handover scenarios.

1.6 Outline

The subsequent sections of this thesis are structured as follows. The chapter titled 'Related Work' delves into the literature related to the domain and technologies employed in this research. Additionally, this chapter offers an overview of essential background concepts and technologies that form the foundation for this study. 'Chapter 3' details the session improvements during heterogeneous networks vertical handovers. Then in 'Chapter 4' we present the base solution for device detection and session improvements during horizontal handovers. Next in 'Chapter 5' we provide robust solution for device detection, considering real-world scenarios. Finally, in 'Chapter 6', we outline the discussion and conclusions drawn in this thesis, the main contributions of our work, and some potential future research directions. The thesis concludes with a 'References' section listing the cited sources.

2

Related Work

This chapter delves into the relevant literature and conducts a survey of associated works pertaining to the technologies and domain under consideration in this thesis. The principal domains and technologies encompassed are the Pervasive speech system, Session Initiation Protocol (SIP), Machine Learning (ML), and audio signalling features. This is necessary to provide the foundation knowledge required to understand the work done in later chapters. This chapter also discusses similar research works which have been carried out in the area of machine learning based session handling and sound source localisation. It also sheds light on the existing gaps in literature and technology that this thesis aims to fill.

2.1 Pervasive Speech

Since 2015 there has been rapid and sustained growth in the number of connected devices around the home; by 2025 it is expected to reach over 20 billion globally [24]. Many of these devices - such as smart speakers, laptops, tablets, smartphones, and electronic portals - can support video and/or audio communication sessions. Whilst this has led to much greater flexibility in how and where we communicate, little effort has been made to unify or simplify the user experience when it is across multiple devices and multiple environments. For example, until recently the only option to switch devices during a call was, to end the session manually on the active device and start a new one on the target device *i.e.* an inherently 'non-pervasive' communications experience, and many of the services related to smart devices remain in siloed ecosystems. However, there are promising signs of the industry moving towards open connectivity standards with the adoption of Matter [25]. This could open up new opportunities for device manufacturers and telecommunications providers alike to deliver features that enhance user experience without being bound to a single brand or service silo [3].

In a move towards 'pervasive' communications, both Amazon and Apple have added features to their respective smart speaker offerings. Amazon has developed the 'group chat' feature [8], [9] for Alexa smart speakers. This uses a broadcast method to join all devices within a specified group to the same call. Apple has taken a slightly different approach by including basic proximity detection in its HomePod smart speaker and iPhones [10]. When a user approaches the HomePod with an iPhone it will handoff audio (for example music or calls) from their iPhone to the HomePod. To transfer audio back again the user taps the HomePod with their iPhone. Both approaches offer incremental improvements to usability in a multi-device setting as basic forms of 'pervasive' communications and still do not take all the advantages of having multiple devices to improve the audio experience. It should be noted that the information provided about these products is sourced from the vendor's online documentation, and the author is not aware with the specific implementation details of the mentioned products.

The authors, Yannick Körber et al. in their research work [26], describe a framework for managing audio playback that supports synchronisation, device selection, and processing that's chosen with respect to user and environmental context. The use of motion controllers (Bluetooth beacons) to detect users and discover devices is discussed (among other commercial solutions for indoor positioning) but the authors deliberately 'abstract from the technical challenges of indoor positioning' by using a manual method for the purposes of experimentation. In their experiment, the authors simulate loudspeaker selection and volume adaptation in a multi-room scenario – essentially test subjects are moved into set positions and the volume of several loudspeakers is adapted manually to try to maintain constant loudness at the different user locations. The authors only address pervasive audio in a music listening context (it does not cover the solution for two-way communications in any way) and the framework does not offer a solution to the device and user orchestration necessary to achieve pervasive communications using anything other than basic user location. The same authors, Yannick Körber et al., in their another work [27], provided a high-level solution for detecting the most suitable playback device for audio playback. This work again does not provide the solution for two-way communication as addressed in this thesis.

To achieve pervasive speech a number of technologies are required, these include the end-delivery system, the method of detecting that a speaker has moved, and signalling systems that can move the necessary audio streams. This thesis will not consider the end systems and will assume that either a mobile device is used in Chapter 3 or that smart speakers are used in Chapters 4 and 5. This thesis will however consider how the switching is decided and how the signalling mechanisms can be made to perform the switching. Consequently, this chapter will first look at SIP as it is the predominant signalling mechanism used for speech and later it will look at methods to detect aspects such as location.

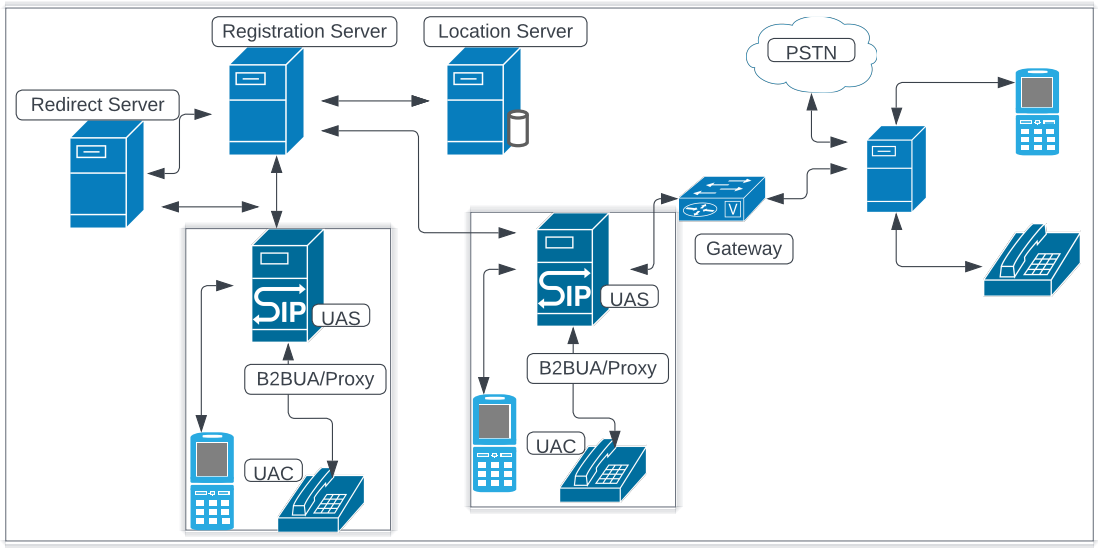

Figure 2.1: SIP Architecture

2.2 Session Initiation Protocol

The Session Initiation Protocol (SIP) [28], [29] has been standardized by the Internet Engineering Task Force (IETF), accepted as a 3rd Generation Partnership Project (3GPP) and widely been used as an application layer signalling protocol in VoIP [29]–[32] and IMS systems [33]. As a signalling protocol, SIP is used to establish, modify and tear down multimedia sessions, which may be audio, video, or even text. SIP is associated with a large number of VOIP applications such as IPTV, instance messaging, online gaming, audio/video conferences, and presence. The key advantage of SIP is, it is independent of the underlying network technology *i.e.* it can be used by both TCP and UDP transport layers. Signalling in SIP is composed of text-based messages. SIP is similar to Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP), where each message can be either a request or a response. The SIP architecture is as shown in Figure 2.1. The main logical entities in SIP consist of Registration, Location, Proxy servers, Redirect servers, and the communicating end-points, namely User Agent Client (UAC) and User Agent Server (UAS). User agents initiate and terminate sessions.

SIP architecture supports a total of 88 messages for session management, 14 are used as requests, and the remaining are the responses. SIP defines an extensible set

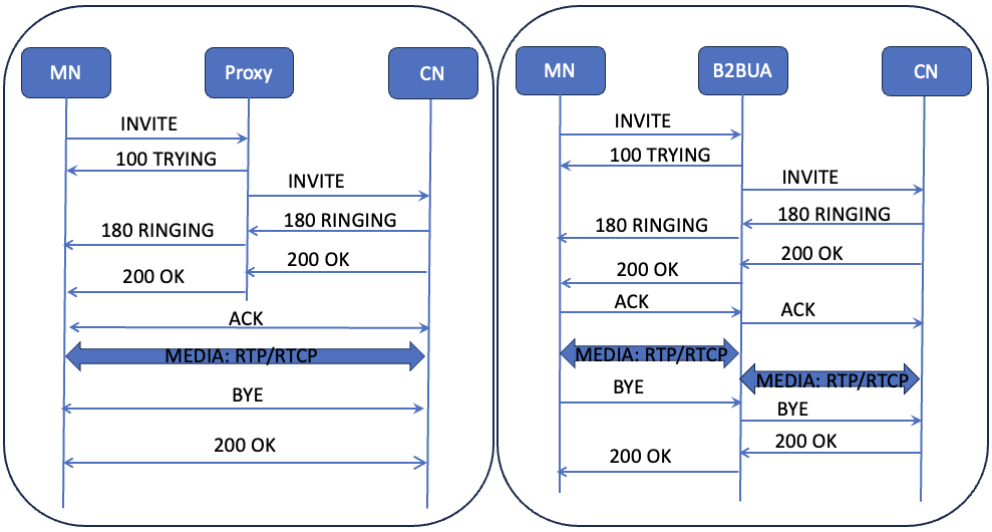

Figure 2.2: SIP Proxy/B2BUA successful call set up message flow

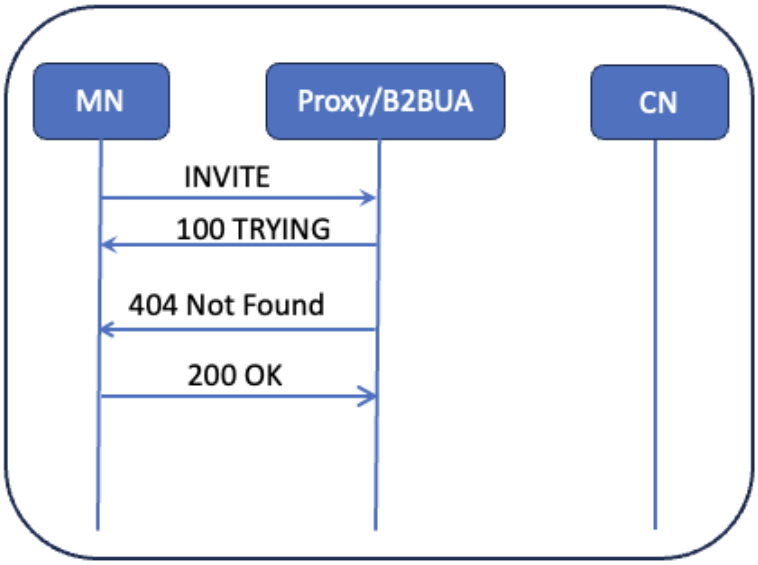

Figure 2.3: SIP Proxy/B2BUA failure call set up message flow

of request methods, currently including in the base specification REGISTER for the registration of a user agent (UA), INVITE to initiate a session, ACK to confirm a session establishment, BYE to terminate a session, OPTIONS to determine capabilities and CANCEL to terminate a session that has not been established yet. SIP also supports, additional extension methods SUBSCRIBE to subscribe for notifications, NOTIFY to notify a particular subscribed event, INFO to send call signalling information, REFER for call transfers, UPDATE to modify the existing session, PRACK to acknowledge 1xx responses and MESSAGE, for instance, messaging services. The basic message sequence

for successful call setup and call failure due to destination not found is as shown in Figures 2.2 and 2.3. There are a minimum of nine messages, flow between the calling and called entity to set up a session, assuming one SIP system between them. This count increases if many SIP systems and also if re-invites are required for call forwarding or for media negotiation. The exchange of several messages over the IP network could cause latency and security issues.

The session itself is typically described using the Session Description Protocol (SDP) that lists media stream addresses, ports, and the encoding. The SIP Registrar receives REGISTER requests from the User Agents (UA) and stores their location information in the Location server for routing the incoming requests to the appropriate network domain.

The SIP server can function fundamentally in two modes: the proxy mode and the Back-to-Back User Agent (B2BUA) mode. The SIP proxy server is used to establish a call session using details from the location server, responsible for routing call requests and assisting in billing operations. A proxy server can be stateless or stateful. A stateless proxy server will only forward incoming requests and responses. A stateful proxy on the other hand will maintain a state for each transaction, that is which requests and responses belong to that transaction. Redirect servers receive requests and respond to the requester where it should send its request.

A B2BUA is a logical entity that receives a request and processes it as a user agent server on one side. On the other side it acts like a user agent client and generates requests. It maintains a dialog state and must participate in all requests sent on the dialog it has established. A B2BUA can also terminate and bridge the media stream and thus have full control over the whole session. This makes B2BUAs well suited for transcoding between two call legs, to hide network internals, for network inter-working, and applying any intelligent session handling as it can have protocol adaptation. This study considers the B2BUA-based SIP server because of its strong features among proxy-based SIP servers.

SIP supports various Application layer mobility such as *terminal mobility*, *personal*

mobility, and service mobility [34], using basic and extension methods. Personal mobility allows a user to be identified by the same logical address even if the user is at different terminals *i.e.* user can register a number of devices (smartphones, smart speakers, PCs) to a single SIP address. The Pre-call *personal mobility* is supported using Call Parallel or Sequential forking. The mid-call *personal mobility* is supported using SIP REFER extension. Terminal mobility allows a device to move between subnets. SIP supports pre-call *terminal mobility* by re-REGISTER of the new terminal and mid-call *terminal mobility* is supported through re-INVITE. Service mobility allows a user to maintain access to their services for example dial lists or address books *etc.*, while moving or changing the devices and network providers. This is supported either by storing the personal information in home servers or carrying the information using SIM or memory cards. This research extends the studies about the *terminal mobility* and *personal mobility* features and proposes solutions for seamless session handovers.

2.2.1 SIP terminal mobility session handling

This research proposes a solution to avoid or minimize the session handover interruptions during handover between the networks *i.e.* during vertical handover, using SIP *terminal mobility* feature. Several studies in the past analysed the session interruption using SIP especially for mid-call *terminal mobility* [35] and improving interruption using proactive handover schemes with SIP extensions [36], [37]. A handover delay comparison study in vertical handover with SIP and without SIP is carried out by authors, Vijayshree et al. in their work [38]. Adapting Machine Learning (ML) techniques in SIP have recently emerged as an alternative to traditional models to improve QoS in SIP. For example, recent work describes using ML to detect attacks against SIP-based services [39]–[41]. However, in this thesis, we are concerned with alternative uses of ML for predicting handover.

Although the author is not aware of any papers directly addressing session handover for SIP using ML, using ML-based user movement prediction in vehicular communication and networks has been studied previously. However, such ML-based user movement

has not been applied to the field of study of this thesis, namely pervasive audio communication, specifically session handover in SIP based communication. In particular, it has been widely studied for the purpose of effective routing towards achieving seamless traffic flow. The work in [42] proposes a supervised ML approach to predict cell and link association duration using cell position and cell load. The motivation for carrying out the work is to anticipate network control decisions in Software Defined Vehicular Networks. However, the paper does not describe how these predictions could provide the control decisions. Consequently, our work describes how the handover interruptions can be reduced using the network transition predictions at the application layer. The work in [43] details an ML approach to predict the handover in heterogeneous networks in an IoT environment. In the study by Memon et al., a Recurrent Neural Network (RNN) incorporating LSTM methods is applied to predict nodes in a Fog network, as detailed in [44]. The machine learning predictive analysis of downlink throughput conducted by Kousias et al. in [45] for Mobile Broadband networks has significantly enhanced the Quality of Service (QoS) for end users in video streaming. These works differ from this research work in that they approach either only position information or addressing routing issues, whereas this work specifically aims at solving the session transition which although related, has important differences.

2.2.1.1 SIP personal mobility session handling

The standard SIP protocol supports *personal mobility* for horizontal handovers during pre-call or even mid-call; SIP *personal mobility* [46], [47] allows, registering multiple devices to one SIP address for session handling. A call session can be initiated on multiple devices with the same SIP address using the existing SIP support of either parallel or sequential call forking [29]. This facility can be used in dynamic session handover between the devices. The use of SIP *personal mobility* has been studied in the past for various smart home applications [48], [49]. This thesis uses SIP *personal mobility* feature for session handling and introduces a novel solution for seamlessly handling sessions during device handovers.

2.2.2 SIP media broadcasting

This thesis considers how media can be sent to multiple end devices. There are multiple ways to do this such as send multiple times by unicast or use layer 3 multicast or layer 4 multipath, the latter two are described below. However this thesis only considers, multiple transmissions in a unicast manner, as other techniques are not widely supported by current networks.

2.2.2.1 Layer 3 multicast

Layer 3 multicast [50] is a method used in networking to efficiently deliver data from one sender to multiple receivers. Operating at the network layer, it uses multicast addresses to manage the distribution of packets to a group of destinations simultaneously, rather than sending separate copies to each receiver. This approach conserves bandwidth and reduces network load by allowing a single stream of data to be replicated at various points within the network. Protocols such as Internet Group Management Protocol (IGMP) [51], for IPv4 and Multicast Listener Discovery (MLD) [52], for IPv6 facilitate the management of multicast group memberships, while Protocol Independent Multicast (PIM) [53], is used to route multicast traffic efficiently. By leveraging these protocols, Layer 3 multicast supports applications like live video streaming, online gaming, and real-time data feeds, ensuring scalable and efficient network performance.

2.2.2.2 Layer 4 multipath

Layer 4 multipath [54] refers to the technique used in transport layer protocols to enhance data transmission efficiency and reliability by simultaneously utilising multiple network paths. This approach, implemented in protocols like Multipath TCP (MPTCP) [55], enables a single data connection to distribute its packets across multiple available routes. By leveraging diverse network paths, Layer 4 multipath improves load balancing, increases bandwidth utilisation, and enhances fault tolerance, as the connection can seamlessly continue over alternative paths if one route fails. This technique is particularly beneficial for mobile devices, where network conditions frequently change, and for ap-

plications requiring high availability and performance, such as video streaming, online gaming, and cloud services. By optimising the use of available network resources, Layer 4 multipath contributes to more robust and efficient data transmission.

2.2.2.3 SIP behaviour to support multicast and multipath

SIP session establishment for supporting Layer 3 multicast or Layer 4 multipath involves a combination of SIP signalling and media transport setup using the Session Description Protocol (SDP) [56]. It leverages SIP-based conference call setup for this functionality support. For Layer 3 multicast, SIP initiates the session by negotiating media parameters through SIP messages containing SDP details, including multicast IP addresses and ports. Once participants agree on the multicast address, routers use protocols like IGMP and PIM to manage multicast group membership, allowing efficient media distribution to all participants. For Layer 4 multipath, SIP negotiates multiple transport paths by incorporating SDP extensions, like the Alternative Network Address Types (ANAT) in SDP [57] or using mechanisms like the Multipath TCP (MPTCP). The SIP signalling coordinates the use of multiple network paths for increased redundancy or throughput, ensuring that the media streams can utilise these paths seamlessly once the session is established. This setup allows SIP to effectively support robust and scalable communication in environments leveraging multicast or multipath networking. SIP involvement in both multicast and multipath functionality, includes multiple session setup sand media broadcast on multiple IP addresses/paths at once. This research proposes parallel session establishment on all the devices, but media transmission on only one device at a time, although can be achieved through layer 3 multicast or through layer 4 multipath, but as stated above, we recommend multiple transmissions in a unicast manner, as other techniques are not widely supported by current networks.

2.3 Machine Learning

We are interested in how the SIP personal mobility session handling, described above, can be automated and we have used Machine Learning (ML) to enable this. ML is a sub-

field of artificial intelligence, the field of study that gives computers the ability to learn without explicitly being programmed. ML approach mainly consists of two main phases: the training phase and decision making phase. One of the machine learning techniques is illustrated in Figure 2.4. At the training phase, machine learning methods are applied to learn the system model using the training dataset. In the decision making phase, the system can obtain the estimated output for each new input by using the trained model. Identifying the key features in the training dataset, called *feature engineering* plays a major role in getting the accurate prediction results from ML methods. Machine learning algorithms are mainly classified into five sub-categories: supervised, unsupervised, semi-supervised, reinforcement and deep learning. Figure 2.5, an extension of the figure from authors, Thomas Rincy et al. in [58], depicts the classification of the machine learning system. Supervised learning is a machine learning task that assumes a function from the labeled training data. In unsupervised learning the data is not labeled, more precisely we have unlabelled data. Semi-supervised learning is a merger of labeled and unlabeled data. In reinforcement learning the software agent gathers from the interaction with the environment to take actions that would maximize the reward. A brief discussion about these algorithms are described in the sub-sections below and further discussions on machine learning theory and its classical concepts are available in the relevant research literature [59]–[63].

2.3.1 Supervised learning

Supervised learning is a labeling learning technique. The labels are mostly called target or ground-truth labels. In these algorithms, the model is trained with the labeled dataset. After training, when a new input is fed into the system, the trained model can be used to get the expected output [64], [65]. Some of the widely-used supervised learning algorithms are Decision tree, Random forest, K-nearest neighbor, Neural networks, Rule based classifiers, Support vector machine, Bayes classifier, and Hidden markov models. Supervised learning is further divided into two major types, Regression and Classification algorithms. Classification algorithms are used to predict categorical or discrete

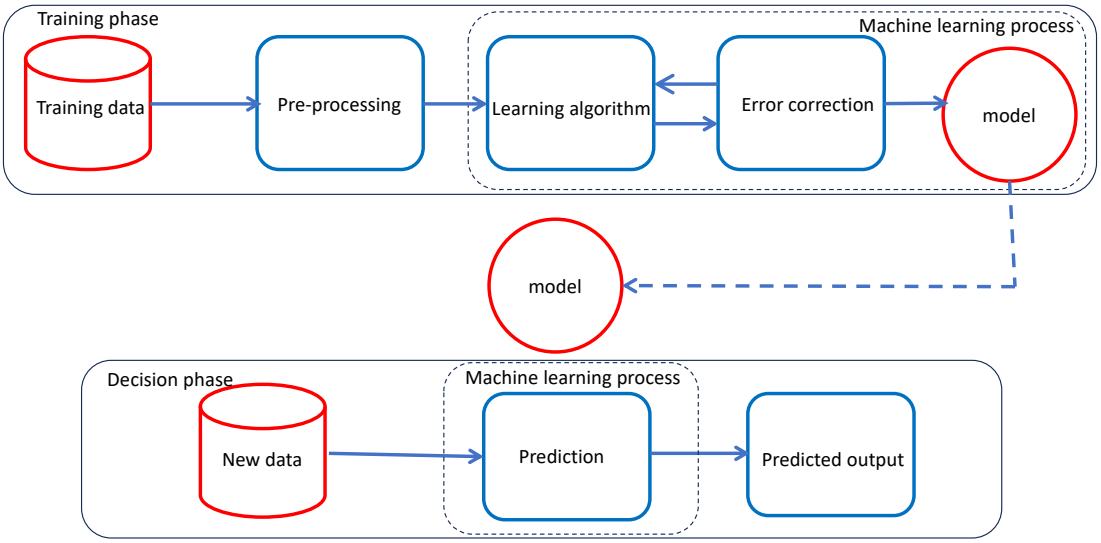

Figure 2.4: Supervised machine learning lifecycle

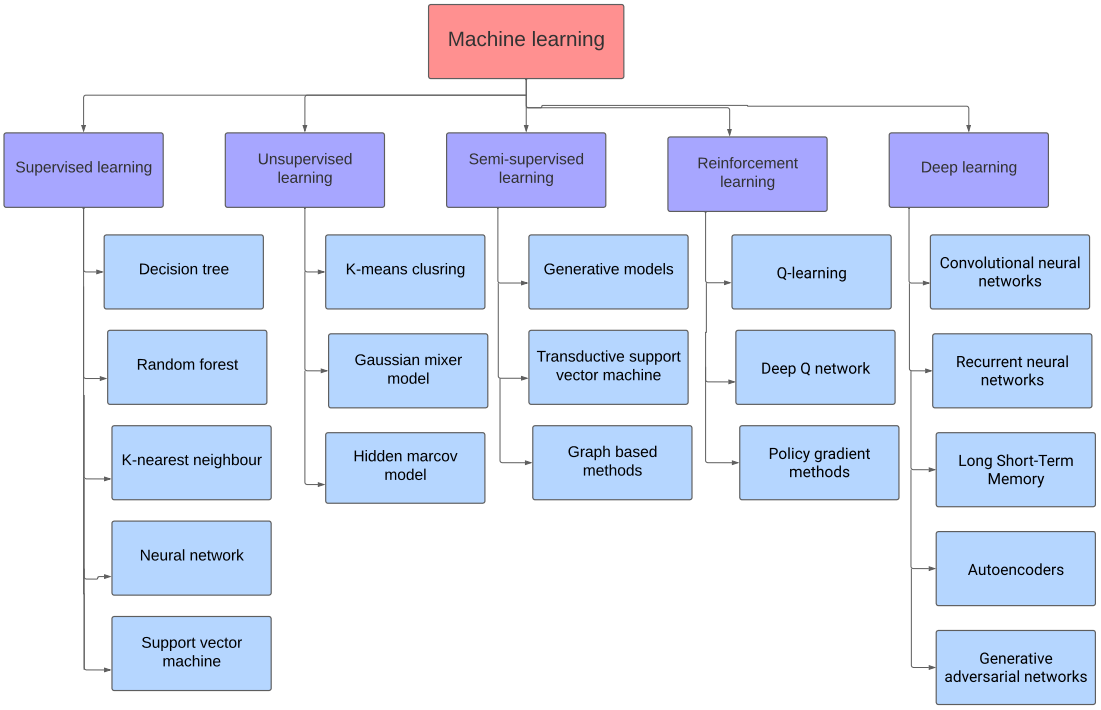

Figure 2.5: Machine learning techniques

class labels and Regression algorithms are used to predict continuous numeric variables. The machine learning technique used in addressing use-case 1, in Section 1.4.1, is based on a comparison of various supervised learning methods. Among them, Random forest demonstrated superior performance for the specified dataset. More comprehensive

information can be found in Chapter 3 of this document.

2.3.2 Unsupervised learning

Unsupervised learning algorithm [66] deals with inputs without labels *i.e.* with no target or ground-truth labels. An unsupervised learning algorithm aims to find patterns, structures, or knowledge in unlabeled data by clustering sample data into different groups according to the similarity between them. Unlike supervised learning, where the algorithm is provided with labeled examples to learn from, unsupervised learning is more exploratory in nature and is often used when the goal is to gain insights into the underlying structure of the data or to discover hidden patterns without predefined targets. Its wide application includes tasks such as clustering, density estimation, and dimensionality reduction [67], [68]. Some of the widely used unsupervised learning algorithms are: K-means clustering, Gaussian mixture model, Hidden markov model [64]. This work does not use this technique because unsupervised learning is well-suited for tasks such as anomaly detection, where a baseline is established, and deviations from this baseline need to be identified. However, it may be less effective in scenarios involving systems with non-linear behavior, where precise classification into known classes is crucial.

2.3.3 Semi-supervised learning

Semi-supervised learning is a combination of supervised and unsupervised learning techniques. It uses both labeled and unlabeled data to build models. In semi-supervised learning, a small portion of the data is labeled, and a large portion is unlabeled. This is mostly used in real-world applications where some label data and some unlabeled data is available for learning. These algorithms make use of the advantages of both supervised and unsupervised techniques. Further details can be referred in [69]–[71]. This technique was not used as the current scope of this thesis, but could potentially be employed for vertical handover predictions as described in Section 3.2.3.

2.3.4 Reinforcement learning

Reinforcement learning [72]–[74] is a training method based on rewarding desired behaviors and/or punishing undesired ones. In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions, and learn through a trial and error process, exploring different actions and observing the consequences. Key components of reinforcement learning include states, actions, rewards, and a learning algorithm such as Q-learning or Policy Gradient methods. The application of reinforcement learning includes robotics, game playing, and optimisation problems. This work does not use this technique because the frameworks do not have feedback mechanisms to reward/punish behaviors; however, it could be a consideration for further work.

2.3.5 Deep learning

Deep learning [75]–[77] is one of the machine learning techniques that focuses on artificial neural networks, specifically deep neural networks with multiple layers. Deep learning models are designed to automatically learn hierarchical representations of data by leveraging complex architectures and large amounts of labeled data. The neural networks, which are part of deep learning models, consist of multiple layers of interconnected nodes, called neurons, that mimic the structure of the human brain. Each layer learns to extract increasingly complex features from the input data, allowing the model to learn very complicated patterns and relationships. Convolutional Neural Networks (CNNs) for image processing, non-linear time-series data predictions, and Recurrent Neural Networks (RNNs) [78]–[80] for sequential data are popular types of deep learning architectures. In this study, CNNs are employed for the detection of smart devices in Chapter 5. Consequently, additional information regarding CNNs is described in the following sections.

2.4 Convolutional Neural Networks

The Convolutional Neural Networks (CNNs) [81], [82], as a leading paradigm in deep learning, excel in time-series analysis, image, and video processing, and have found utility

in a broad spectrum of domains, encompassing applications such as human activity recognition and speech recognition. They operate by employing convolutional layers to automatically learn and extract intricate features from data. Convolutional layers in CNNs typically consist of multiple filters with varying kernel sizes, enabling the network to extract a hierarchy of features at different levels of abstraction. Filters and kernel sizes play a fundamental role in feature extraction and pattern recognition. 1DCNN is one of the types of CNNs architecture that is particularly effective for processing sequences or time-series data. While traditional CNNs are widely used for image data, 1DCNN are designed to capture patterns in one-dimensional sequences, such as those found in audio signals and time-series data. In this study, the 1DCNN technique is employed for device detection, as discussed in Chapter 5. Further elaboration on the majority of key components within CNNs is provided, with specific mention of certain components utilised in the research detailed below:

2.4.1 Convolutional layer

Convolution operation: CNNs use convolutional layers to perform convolution operations on input data. Convolution involves sliding a filter (also called a kernel) over the input data, element-wise multiplying the filter values with the overlapping input values, and summing up the results. This operation helps the network learn hierarchical features.

Filters and Feature maps: Filters act as feature detectors, and the output of the convolution operation is a feature map. Multiple filters are employed to capture various features in the input.

2.4.2 Activation function

Rectified Linear Unit (ReLU): The output of the convolutional layer is often passed through a non-linear activation function like ReLU. For a review of all currently supported activation functions, refer to the work by Rasamoelina et al. in [83]. ReLU introduces non-linearity by setting all negative values to zero, allowing the network to

learn complex relationships in the data. Without this non-linearity, a neural network would essentially behave like a linear model, regardless of the number of layers, significantly limiting its ability to capture intricate patterns. By incorporating non-linearity, ReLU allows the network to learn and represent a much broader range of functions. Unlike other activation functions, such as sigmoid and tanh, ReLU does not saturate in the positive domain, which helps mitigate the vanishing gradient problem to some extent. This is crucial for maintaining a strong gradient and ensuring effective learning during backpropagation. Additionally, ReLU is computationally efficient due to its simple thresholding operation, making it faster to compute than more complex activation functions. ReLU also introduces sparsity into the network by outputting zero for negative inputs, which can help prevent overfitting and improve model efficiency.

2.4.3 Pooling layer

Pooling operation: Pooling layers are used to downsample the spatial dimensions of the input data and reduce its computational complexity. Common pooling operations include max pooling (selecting the maximum value from a group of values) and average pooling.

Spatial hierarchical features: Pooling helps retain the most relevant information while discarding less important details. It also aids in creating spatial hierarchical features.

2.4.4 Fully-Connected (FC) layer

Flattening: After several convolutional and pooling layers, the data is flattened into a one-dimensional vector. This vector serves as the input for one or more fully connected layers.

Dense layer: A dense layer is a fully connected layer where each neuron receives input from all neurons of the previous layer, enabling the model to learn complex patterns by adjusting weights through backpropagation.

Class scores: Fully connected layers are responsible for making predictions based

on the learned features. The output represents class scores in classification tasks.

2.4.5 Softmax layer

Softmax activation: In classification problems, the final layer typically applies the softmax activation function. Softmax converts the raw class scores into probabilities, making it easier to interpret the network's predictions.

2.4.6 Loss function

Loss functions, also known as objective functions or cost functions, play a crucial role in training CNNs. The primary purpose of a loss function is to quantify the difference between the predicted output of the model and the true target values. Some of the standard or pre-defined loss functions are described below:

Mean Squared Error (MSE): Calculates the average squared difference between predicted and actual values, mainly used to solve the regression problems.

Binary Cross-Entropy loss: Measures the binary classification error between predicted and true labels, mainly used to solve binary classification problems.

Categorical Cross-Entropy loss: Extends cross-entropy to multi-class classification problems, mainly used to solve multi-class classification tasks.

Sparse Categorical Cross-Entropy loss: It is similar to categorical cross-entropy but suitable when the target values are integers.

Custom loss function: A custom loss function refers to a user-defined mathematical function that quantifies the difference between the predicted output of a model and the actual target values. Unlike standard or pre-defined loss functions provided by machine learning frameworks, a custom loss function is specifically tailored to address the requirements or challenges of a particular task or application. Custom loss functions are integrated into the training process of machine learning models. The effectiveness of a custom loss function is validated through experimentation. This involves training the model with the custom loss function and evaluating its performance on validation datasets. Fine-tuning will be necessary to achieve optimal results. This crucial fea-

ture serves as the foundation for delivering a robust solution for device detection, with additional information accessible in Chapter 5.

2.4.7 Backpropagation and optimization

Backpropagation: The network adjusts its weights and biases during training using backpropagation. This involves computing gradients of the loss with respect to the network parameters.

Optimization algorithm: Optimization algorithms (e.g., stochastic gradient descent) are used to minimize the loss function by updating the network's parameters.

2.4.8 Regularization techniques

Dropout: Dropout is a regularization technique where randomly selected neurons are ignored during training to prevent over-fitting.

Batch normalization: Batch normalization normalizes the inputs of a layer, making training more stable and improving generalization.

2.5 Audio Signal Features

This research investigates and evaluates various audio signalling features. Rather than creating new types of features, this thesis draws upon those that are commonly used as described well by Mitrović et al. [84]. These features include Magnitude Squared Coherence (MSC), Signal Magnitude (A), Mel Frequency Cepstral Coefficients (MFCC) as potential features for device detection. An overview of these features is presented in the following sections, with more in-depth details provided in subsequent Chapters 4 and 5 where their utilisation is discussed.

2.5.1 Magnitude squared coherence

Magnitude Squared Coherence (MSC) is widely used in speech recognition, audio analysis, and detection [85]. It is a signal processing technique that indicates how well two time domain signals match one with the other by tracking linear dependencies in

their spectral decomposition. The values of MSC range from 0 to 1, where 0 indicates no correlation or coherence between the signals, and 1 indicates perfect correlation or coherence. MSC accounts for the frequency distribution of the signals, making it suitable for analysing signals with varying spectral characteristics. This signal feature is pivotal for device detection, and additional insights into its utilisation are described in Chapters 4 and 5.

2.5.2 Signal magnitude

Signal Magnitude (A) represents the magnitude of a signal. It considers only the positive magnitudes and removes the negative values of the signal. The absolute value operation can be applied to various types of signals, including continuous-time signals, discrete-time signals, and digital signals. It is a simple yet essential mathematical operation that helps extract meaningful information and simplify audio analysis and processing tasks. Additional information on the application of this feature is elaborated in Chapter 5.

2.5.3 Mel frequency cepstral coefficients

Mel Frequency Cepstral Coefficients (MFCC) is one of the audio features that can be used for audio analysis and detection [86], [87]. They capture important information about the spectral characteristics of the signal, focusing mainly on frequency components. The computation of MFCC involves a series of steps. Initially, the signal is segmented into frames. Subsequently, windowing is applied to mitigate disruptions in each frame. Fast Fourier Transform (FFT) is then employed on each frame to compute the magnitude spectrum. Following this, a filterbank consisting of triangular filters is applied to the magnitude spectrum. These filters are spaced according to the Mel scale, a perceptual scale of pitches. The Mel frequency scale, being logarithmic, more accurately reflects the human ear's response to varying frequencies. The next step involves taking the logarithm of all filter bank energies. Finally, the Discrete Cosine Transform (DCT) is calculated for the logarithm of filter bank energies. The resulting DCT output yields the Mel Frequency Cepstral Coefficients (MFCC). Further details regarding the application

of this feature, along with other signalling features, are elaborated in Chapter 5.

2.6 Sound Source Localisation

This thesis builds upon some of the concepts of Sound Source Localisation (SSL) for device detection, although we aim to avoid using precise localisation which typically requires either accurate calibration or external systems as described below. The subject of SSL has been studied for various audio signal processing applications such as robotics, video conferencing, smart home systems, speech enhancement, ocean engineering, and military systems. SSL involves estimating two parameters of the acoustic signal, namely direction of arrival (DoA) and distance of arrival estimation. Both classical and learning based solutions have been proposed to solve the SSL problem. Classical approaches [88], [89] require *a priori* knowledge of the acoustic source environment *i.e.,* physical room characteristics such as the room dimension, room impulse response, surface area of the walls *etc.* Hence, learning based approaches have recently been proposed for localising using sound distance estimation without these classical measurements [90]–[93]. Some of the audio features such as Magnitude Squared Coherence (MSC), Signal magnitude (A), Direct-to-Reverberant Ratio (DRR), Mel Frequency Cepstral Coefficients (MFCC) *etc.* are considered to be the key features in learning based methods for sound localisation. The work in [91] is a comparative study on calculating the distance of arrival based on various acoustic source distance parameters and concludes that the MSC gives the best method for distance calculation. The work in [94] describes the use of MSC to calculate the DRR, a useful parameter in acoustic applications. The work in [90] proposes a Gaussian mixture model (GMM) using magnitude squared coherence feature from a binaural input for a distance of arrival estimation. Their work uses the MSC to estimate distance using *a priori* calibration. While these methods provide good motivation for using the MSC for distance estimation following calibration in static environments, they do not solve the problem of generic device switching where calibration is not available. However, the finding that the MSC is a highly useful metric gives motivation for our work in Chapter 4.2.2 using the MSC for finding the most suitable device without calibra-

tion. Furthermore, these prior works do not consider generic techniques for optimising switching between devices where the position of the devices is not closely controlled, and do not address the signalling required to enable intelligent switching for smart devices. However, relying solely on MSC for sound source localisation has limitations. Consequently, this research delves deeper into the analysis of deep learning techniques for device detection. As described in Section 2.4, CNNs, one of the dominating paradigms in deep learning [95] have exerted a substantial influence on research across diverse domains, such as Human activity recognition [96], [97], Image classification [98], various time-series prediction applications [99]–[102] *etc.* In particular, the 1DCNN [81] has achieved good results in processing time-series multivariate audio signal data. These previous works provides motivation for this research in utilising a 1DCNN to provide robust smart device location for pervasive speech communication.

2.7 Acoustic Room Simulation

This thesis uses acoustic simulation to test and evaluate the research. Pyroomacoustics [16] is an open-source Python library that provides libraries for simulating room acoustics and sound propagation. It was selected as it provides an accurate model while balancing this with the computational requirements by using a highly optimised C-based library. It can be used to model and simulate the behavior of sound sources in a given room environment. Using libraries provided by Pyroomacoustics, a virtual room with desired room dimensions, sound sources and microphones in desired positions can be modeled. The libraries allow to place microphones, each with one or multiple microphone arrays, at specific locations within the room and simulate their responses to sound sources. The room can also be simulated with various Reverberation time (RT60) values [103], [104], which can simulate the sound reflections and walls with different absorption coefficients which are helpful in deep analysing room acoustics. The library can use both image source and ray tracing simulation. As the thesis uses rectangular rooms, the image source model was selected as it is the most efficient for this type of room, without sacrificing any accuracy.

2.8 Research gaps

Pervasive speech communication seeks to deliver a seamless audio and/or video experience based on the situational context, such as the preferred end device or the best available network, both before and during a call session. After conducting a comprehensive literature survey, this research identifies significant gaps in achieving a pervasive speech experience for users. Currently, if a user wants to switch devices during a call, they generally need to manually end the session and initiate a new one on the target device. There are still session latency and media disruptions during transfers between devices and also between the networks. The overall research literature survey concludes that:

- There are promising signs within the industry toward addressing this issue and moving toward providing solutions for a 'pervasive' communication experience. Companies such as Amazon and Apple have introduced features like 'group chat', 'HomePad' *etc.* solutions to their smart speaker offerings. Although these solutions bring us closer to achieving a 'pervasive' experience, they still fall short of meeting the requirements of pervasive speech, especially in scenarios where two-way communication is not feasible with the current industry solutions.

- In academia, the author did not encounter any significant research specifically focused on the pervasive speech experience.

- The existing SIP session handovers between the networks i.e. *vertical handover* encounters session latency and media disruptions.

- The existing SIP session handovers between devices within the same network domain, known as *horizontal handovers*, experience session latency and media disruptions.

This research aims to address these challenges by proposing solutions to enable more fluid and uninterrupted transitions, ultimately enhancing the overall user experience in

pervasive speech communication. The following chapters in this research details the propose solutions for the identified gaps.

2.9 Conclusions

In conclusion, the exploration of related work in this thesis has provided valuable insights into existing research and methodologies in the fields of SIP signalling, machine learning, audio signal features, and network handovers. This chapter provided a high-level view of the SIP protocol, its architecture, and its special features which are considered for session handovers in this thesis. This chapter also highlighted the various machine learning techniques, their applications, and details about Convolutional Neural Networks, as this is the key technique used in this work. This chapter also reviewed research related to Sound Source Localisation, as this research builds upon some of its concepts for device detection. The identified gaps, trends, and contributions outlined in the related work not only inform the rationale for the current investigation but also lay the groundwork for addressing unanswered questions and advancing the knowledge in these domains.

3

Seamless Vertical Handover using Machine Learning

This work considers how session initiation protocol systems can operate in a pervasive communication scenario, in particular when there is mobility causing vertical handover between delivery networks. This work uses a machine learning based approach to predict users' transitions and thus overcome interruptions in speech due to signalling handover. As pervasive communication systems are not currently available for measuring the performance of the solution, the work is carried by using commonly available dataset for vehicular mobility to assess likely handover performance. The results show that prediction can reduce the more concerning interruptions (>2s) due to vertical handover events by more than 99.9%.

3.1 Introduction

The core concepts pervasive speech system and SIP have been introduced in Chapters 1 and 2, this chapter details the work related to session handling during network transitions. The main contributions of this chapter are towards understanding likely call interruptions for SIP based speech when transitioning between heterogeneous networks *i.e.* vertical handover, and improving on this problem using a Machine learning (ML) approach to provide session continuity with no, or little interruption. This work uses a data-driven approach by using existing mobility data from a real-world vehicular mobility dataset to train the supervised machine learning model for predicting the next network transitions. Using this prediction, a proactive network SIP session handover can be triggered and with this advanced session establishment onto next likely network attachment point, speech can be transmitted in advance to allow a *make-before-break* handover in heterogeneous networks in a manner that we are familiar with from homogeneous networks, such as LTE. While this work uses a vehicular dataset for evaluating the problem and solution, in the absence of existing real-world pervasive communication application data, the approach is designed to operate across any type of IP networks where IP connectivity may change during the conversation. Of course, in a current vehicular scenario, a user tends to use a single network such as a 4G/5G mobile network, in such a case that network handles all the mobility. Here, we are looking to a future application which moves seamlessly between network types (For instance, as a user transitions between being connected to 5G Long Term Evolution (LTE) in their car and connecting to Wi-Fi at home, or vice versa). This work has been published in a conference paper in IEEE International Conference and Expo on Real Time Communications at IIT (RTC), 2021 [C1].

3.2 Proposed Method

The subject of proactive vertical handover using SIP has already been studied for some time in the past. However, the techniques for proactive handover were based on soft

handover ("make-before-break") in link-layer in heterogeneous networks [37]. In our study, we are considering the handover execution at the application layer using SIP and the handover decision is supported through ML processing by predicting the next likely, attachment point and, optionally, the estimated time to move to this next attachment point. It should be noted here that the bulk data processing power of ML is carried out at the training stage and that the processing at the prediction stage is relatively simple and this is helpful in handling a large number of handovers. The ML is carried out in a central server for the proposed pervasive communication SIP application. A suitable server is the SIP Registration server [30] which receives updates with user registrations and change of point of attachment. This needs to identify only the local attachment network that a device is connected to, for example, different Wi-Fi hot-spots; crucially, the ML does not need actual geographical location data. While geographical data maybe available in some horizontal handover systems, such as LTE/5G, this information is not generally available in heterogeneous networks which is the focus here. In this work, we are using a supervised learning approach as we are using an existing dataset *i.e.*, mobility data learned from a previous day with *ground-truth* labels. In practice, it is more likely to be implemented as a semi-supervised algorithm with the model refined on a daily or weekly basis from past data. The ML uses information about attachment networks and times between changes of attachment to predict the next likely network attachment and the time to change to this new attachment. Thus, SIP session continuity can now be provided through the prediction from the ML using the features of point of attachment and duration of attachment.

3.2.1 SIP session continuity

We have given, above, a brief high-level overview of the mechanism proposed in this work, here we discuss one possible implementation of the mechanism in the context of the SIP signalling. Vertical handover can occur during pre-call or during mid-call. This work considers the mid-call mobility scenario, where session and media update is expected for the existing session. A graphical representation of proposed solution is as

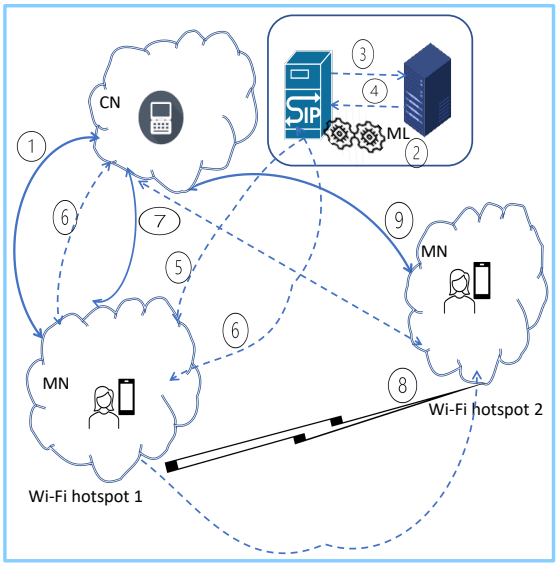

Figure 3.1: SIP vertical handover-proposed solution

shown in Figure 3.1. The description below will introduce the solution in more detail and explain some of the new concepts.

When the mobile node moves from one network to another it first contacts the Registration server[1] to de-register the previous IP address and register the new address at the new point of attachment. When the Registration server coordinates with the SIP Location Server to register this new address it queries the ML predictor using: the previous point of attachment, the new point of attachment, and the time interval between registration of the previous and new point of attachment. From this query, the ML predictor then estimates the next point of attachment and, optionally, an estimate of the time to change to the next point of attachment. Consequently, an agent at the new attachment point is notified to expect a likely new attachment in the future and replies back to the Registration server with the future IP address. Finally, the mobile node is notified, via a SIP NOTIFY message, of the next likely point of attachment.

The existing SIP signalling dialog for a mid-call change of network attachment is shown in Figure 3.2. It sends a re-INVITE to its corresponding node with its new IP

[1]Note, we use a singular form for a registration server; however, in practice, multiple actual servers will be used, in this case, the daily updated ML model would need to be synchronised across the registration servers.

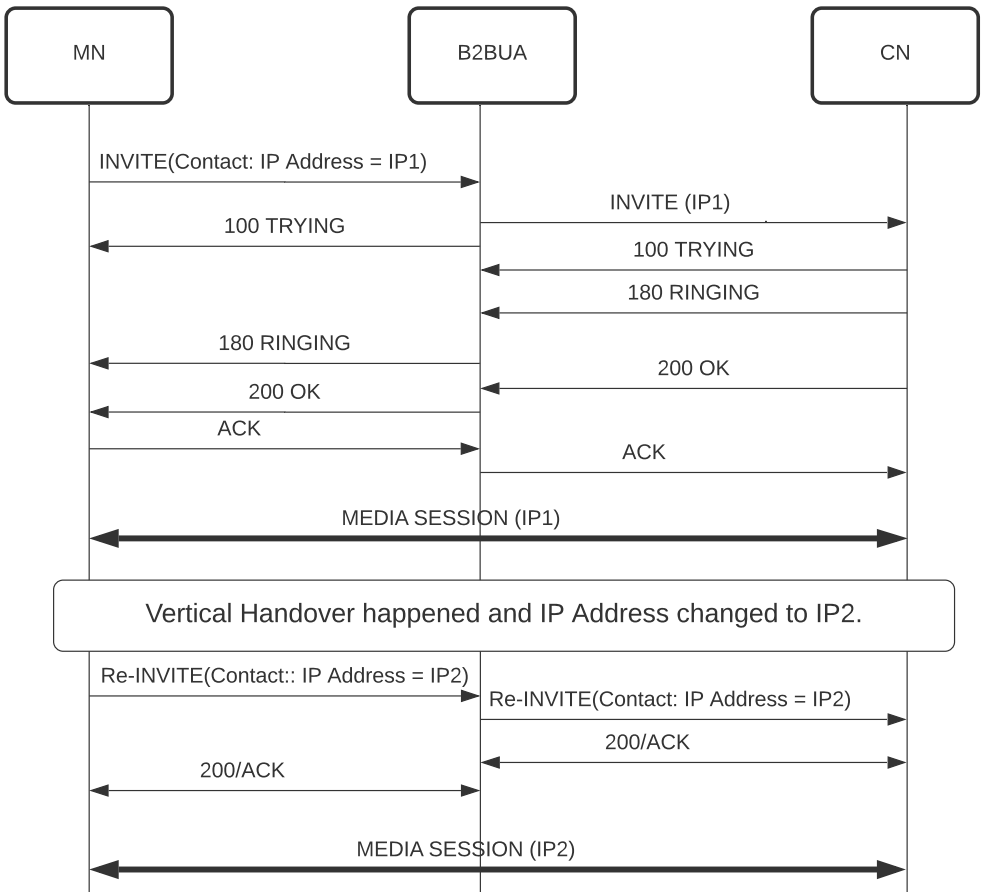

Figure 3.2: SIP mid-call flow when there is an address change due to handover between a mobile node (MN) and a corresponding node (CN)

address as its new Contact [29]. There are challenges using re-INVITE in the existing dialog, due to the handover delay, SIP re-session overhead, Registration/Location overhead and SIP message size; this is not helped by the text-based nature of SIP. Crucially, media handover cannot happen until the signalling messages are complete and consequently, there are interruptions to the speech during this handover when using an existing SIP-based mechanism. Consequently, this work will propose modifications to the signalling process.

In this work it is proposed that the media handover interruption is removed using proactive media session handling using the previously described ML to predict the network mobility so that media delivery is established before the physical move *i.e.*, *"make before move."* The existing SIP standards [29] do not support, session and media on two IP addresses. However, a session handover approach called *bicasting* using a SIP

Extension is proposed by Toktam *et al.* [37] in the context of session setup using *"make-before-break"*. While the original bicasting approach [37] does not solve all the problems in a heterogeneous network, as it was designed to operate on the Link-layer only, the signalling and media delivery can be re-utilised in this work. Specifically, bicasting is used to maintain media continuity by sending media to both the current and the next predicted point of attachment. The session call flow using bicasting, when the corresponding node supports the Require:HANDOVER extension, is as shown in Figure 3.3. In the case that the corresponding node does not support the Require:HANDOVER extension, then solutions like a back-to-back user agent (B2BUA) can be used as a bridge for both session and media as proposed in [36] for backward compatibility.

3.2.2 Mobility dataset

To analyse our mobility prediction for a pervasive communication application we would ideally need realistic data from a communication application operating over heterogeneous networks, for example moving between Wi-Fi hotspots or between an LTE mobile phone and a Wi-Fi smart speaker. In the absence of realistic data from such a future application, we have used a vehicular mobility dataset. The main difference between the vehicular mobility dataset and the expected pervasive communication application is that the former deals with a greater number of networks with fewer handovers while the pervasive communication application is expected to deal with only one or two networks (e.g. a user entering from outside to home and transition happening between a phone in LTE network to the smart speaker in Wi-Fi at home) but more frequent handovers.

The handover prediction example used for this work is based upon actual mobility data from the city of Cologne (described in more detail in Section 3.2.2.1 below). Paired with this we used actual network locations (Section 3.2.2.2 below) from Cologne to simulate where these actual users were connected as they moved in their vehicles. While it would be possible, from this first dataset, to use the vehicle location for the purpose of prediction, this is unrealistic in a practical networked application scenario as network elements do not normally have access to this information. Consequently, for the ML

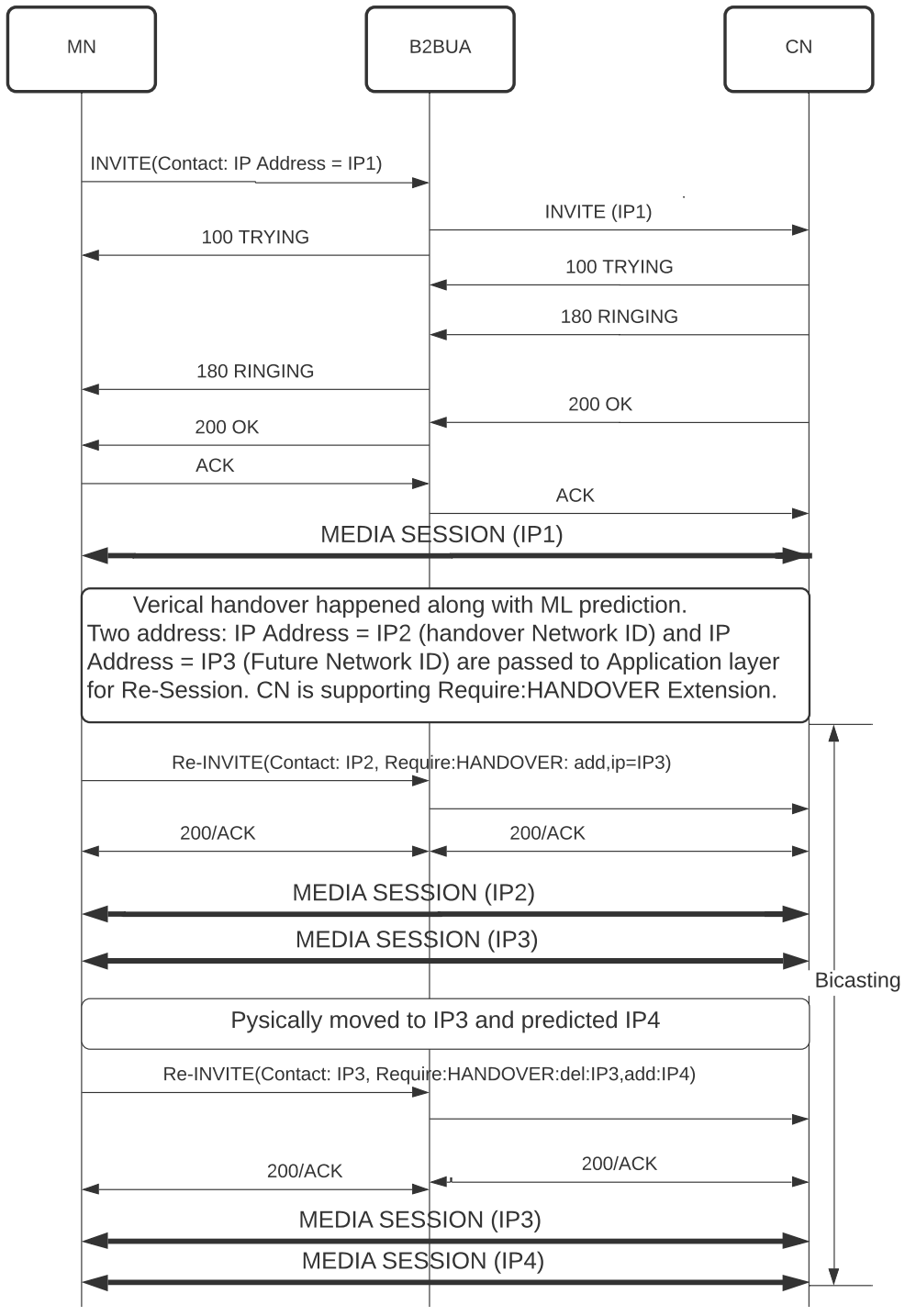

Figure 3.3: SIP mid-call flow with two IP addresses - BiCasting: Using SIP Extension

prediction we used only the previous network attachment identifier (ID), the current network attachment ID and the duration in the previous network. This was then used to predict the next network ID and duration at the current network ID *i.e.,* the time until the next transition. These features were determined from a second network attachment

Table 3.1: Vehicular mobility dataset parameters

Parameter	Description
Vehicle Id	A unique vehicle identifier
Vehicle position	X,Y coordinates of the vehicle position, used for deriving the vehicle's association to nearest wireless access point
Wireless access point position	X,Y coordinates, that can be used to represent a network attachment ID
First Timestamp	Time of the vehicle joining a new network attachment
Second Timestamp	Time of the vehicle leaving a network attachment

Table 3.2: ML features generated from vehicular mobility dataset parameters shown in Table 3.1

Feature	Description
Previous network ID	Previous network attachment ID
Previous attachment duration	Duration that a device was connected with the previous network
Current network ID	Current network attachment ID
Future network ID	The next predicted (or actual) network attachment ID
Current network duration	Duration predicted (or actual) that this MN will be connected with the current network before moving to the Future network ID

dataset that was generated from the vehicular mobility dataset as described below.

3.2.2.1 Vehicular mobility dataset

The mobility trace contains vehicle mobility information with a 1-second granularity for an anonymised vehicle identifier, its position on the two-dimensional plane (X and Y coordinates in meters), and its speed (in meters per second) taken over a 24 hour duration in the greater urban area of the city of Cologne. This trace contains the mobility behaviour of more than 650,000 vehicles of different types. The generation of this realistic data set is clearly described in research papers associated with the dataset [14], [15]. The parameters of this feature set are described in Table 3.1.

3.2.2.2 Network attachment dataset

As noted earlier, it is not realistic for network elements to know precise MN locations and speeds. Consequently, to generate the features for the ML we assume only knowledge of network attachment. To simulate this for a future pervasive communication application we used the base station location information from public German databases in 2012 using the same coordinate system as the vehicle positions. Applying a Voronoi Tessellation on base station locations provides a good estimate for each network attachment point in the region [105]. Using this approach we generated the network attachment dataset directly from the realistic vehicular mobility dataset. This resulted in the features shown in Table 3.2.

3.2.3 Machine learning approach

There are many machine learning algorithms and approaches, and detailed descriptions are given widely in the literature [106], [107]. Analysis of the problem described in this work means that a supervised (or rather semi-supervised) approach is appropriate. Information about supervised machine learning is explained in Chapter 2, and a comparative survey, particularly focusing on this algorithm, is provided by [108], [109]. This work compared a number of supervised machine learning techniques [108] including: linear regression, naive Bayes, decision trees, multi-layer neural networks, random forest (RF), and k-nearest neighbour (KNN). After comparison, RF [110] and KNN [111] were found to perform the best and are the only two shown in this work. While comparing in detail the reasons for these algorithms success is beyond the scope of this work, it can be noted that the problem is highly non-linear (mapping arbitrary network ID histories); consequently, rule based approaches like random forests can perform well in these cases. Additionally, random forests supports both classification and regression from the same input dataset. It was noted that random forests can handle large datasets efficiently, although this was less critical for this application. As noted at the start of Section 3.2 the ML would be implemented as a semi-supervised approach in practice by using data from a previous day; or, day of the week, as mobility patterns vary weekly.

3.3 Results

To evaluate the performance of the proposed handover solution, the network attachment dataset was generated as described in Section 3.2.2. The key metrics of the work are to analyse the call interruption time caused by SIP session handovers. However, several options were considered in the application of the ML prediction and these have various impacts in the performance of the proposal in this work. The results were generated according to the following scenarios:

no prediction (NP) the default using standard SIP handover with no bicasting;

next network prediction (1P) bicasting speech to both the current network and the predicted next network

next network prediction to two most likely (2P) bicasting (or rather "tricasting") to the current network and the two most likely predicted next networks

next network prediction with regressor (Px) send speech to the current network and then bicast to the predicted network x s before the regressor predicts the transition

In the above, the regressor was introduced in the Px scenarios to reduce the time that duplicate traffic is sent to the next network. In practice we will see that, while the regressor has good median prediction, it has a relatively wide variation around this median performance which means that if the prediction is too long there will be an interruption in the call as the transition happened earlier than the predicted transition. This is shown in Figure 3.4 for two hypothetical transitions A and B. If the raw regressor estimation estimated the transition too late (top dotted line) then there will be a break in audio as it moves the audio after the transition to B. The estimation error margin shows five different error margins with P5 (5 seconds) *etc.* P5 is a large error margin and means we can be confident that it will transition in time, but there will be a longer period of bicasting leading to less efficient network transport. Alternatively, P1 is too low and leads to a late transition. Determining an optimal error margin x is likely to be system-dependent. Here a range of values were added to the regressor predictor and

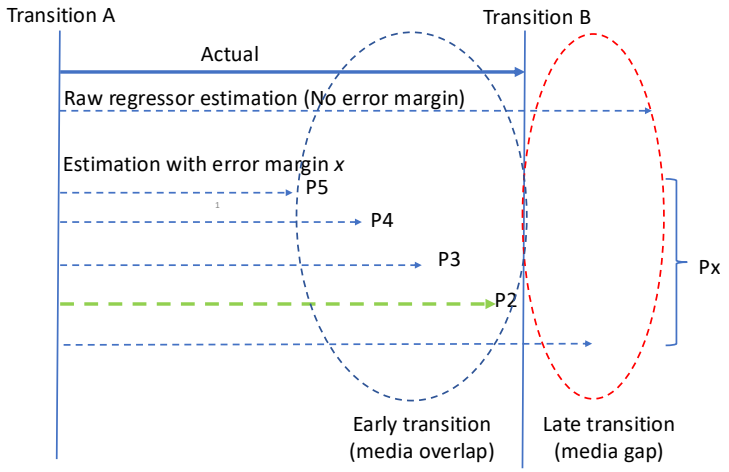

Figure 3.4: A diagrammatic explanation of regressor error margins: A late transition estimate leads to a media gap, while a larger margin, such as P5, results in more audio waste but no media gap during the transition. In this figure, P2 achieves seamless media transition with minimal audio waste

these are shown in the results as Px between 2 and 32 seconds.

3.3.1 Calculating the SIP session establishment time

We have performed measurements of SIP call session establishment latency using a SIP test-bed under low load conditions, running on a local system *i.e.* such that additional processes, link latencies or CPU limitations were not significant.

The session successful establishment latency for a single user repeated 100 times is shown in Figure 3.5 indicating that median times for call establishment (or re-establishment after handover) are in the order of 280 ms. In practice, there may be additional factors that will increase this latency namely, additional proxies, re-registration process, load on the proxies, distance from user to proxies *etc.* Our measurement of 280 ms is comparable with others that have measured SIP signalling latency [112]. Also, we are assuming this session setup time includes the time required for re-registration and location updates, again in practice this might cause higher latency. For the modelling results, we sampled following the measured latency distributions shown in Figure 3.5 so

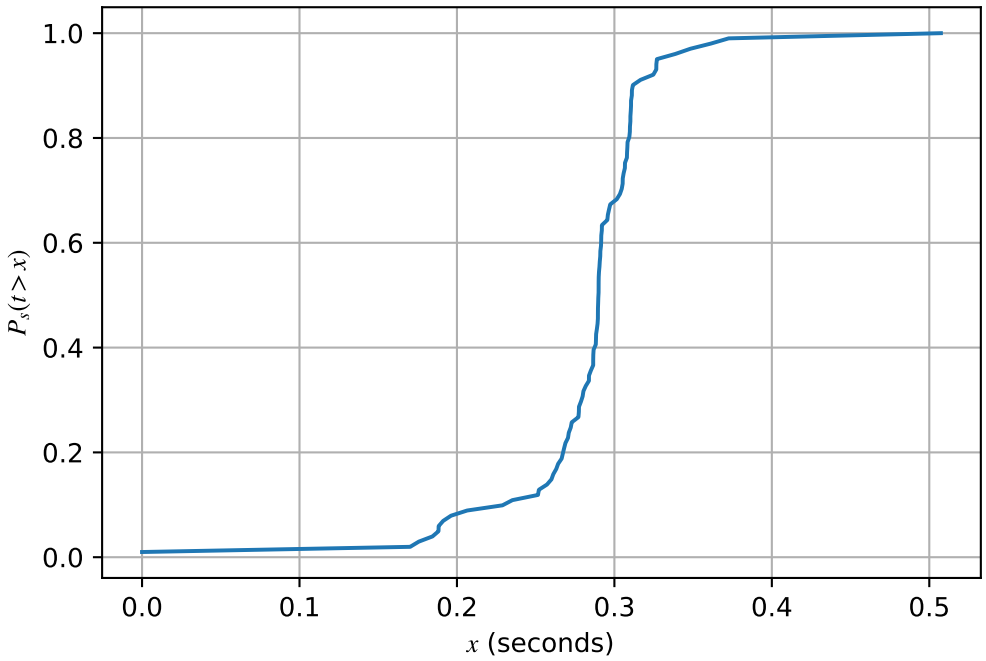

Figure 3.5: Empirical cumulative distribution function showing SIP call set up latency t

that the results were based upon these real latency values.

3.3.2 Calculation of network attachment dataset

We have analysed handovers from approximately 650,000 vehicles across a total of 4.5 million handovers from one network attachment point to another attachment point. A vehicle having at least 2 handovers was considered for processing *i.e.,* static "mobile" nodes were excluded. We assumed each handover is a vertical handover. While the later may not be realistic for a true vehicular case it acts as a useful proxy of generated events for a future pervasive communication application. In order to derive the vehicular transitions, we have created a processing algorithm with inputs: the generated network IDs from the Network attachment dataset, vehicle ID, and timestamps of transitions to determine network attachment duration. The vehicle's association with a nearest network attachment point was determined from the vehicular mobility dataset using the k-d Tree algorithm [113].

Table 3.3: Performance of the Classifier

	1 prev. net.		2 prev. net.	
Metric	RF	KNN	RF	KNN
Classifier Accuracy	0.793	0.785	0.937	0.929
Classifier F1 score (macro)	0.747	0.737	0.904	0.907
Classifier F1 score (micro)	0.793	0.785	0.937	0.929
Classifier F1 score (weighted)	0.790	0.782	0.936	0.938

Table 3.4: Performance of the Regressor

	RF	KNN
Regressor Median Absolute Error	7.80	8.049
Regressor Median Relative Error	1.8%	1.9%
Regressor Coefficient of determination R^2	0.299	0.296

3.3.3 Machine learning performance

In order to predict the next, likely, attachment point and the estimated time to move to this next attachment point, we have trained our model with the history of vehicle transition *i.e.* previous network ID, duration in the previous network, and current network ID. The estimated time to move to the next attachment point is predicted with RF and KNN regressor algorithms and the next likely network attachment point prediction is using RF and KNN classifier algorithms with performance results shown in Tables 3.3 and 3.4. Note that due to a large number of classes, it is not straightforward to analyse the ML performance using common metrics such as a confusion matrix or recall/precision; however, a small portion of the confusion matrix is shown in Figure 3.6 which shows a strong diagonal component which gives some confidence in the performance of the ML. Considering the nature of our model as a *multi-classifier*, we have calculated both the classifier accuracy: representing the ratio of correct predictions to total predictions, and F1 scores (*macro, micro, and weighted)*, representing the scores of precision and recall. For the regressor we show median absolute and relative error and the coefficient of determination, R^2, showing the performance of the regressor model to predict the duration of attachment:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)}{\sum(y_i - \bar{y})} \tag{3.1}$$

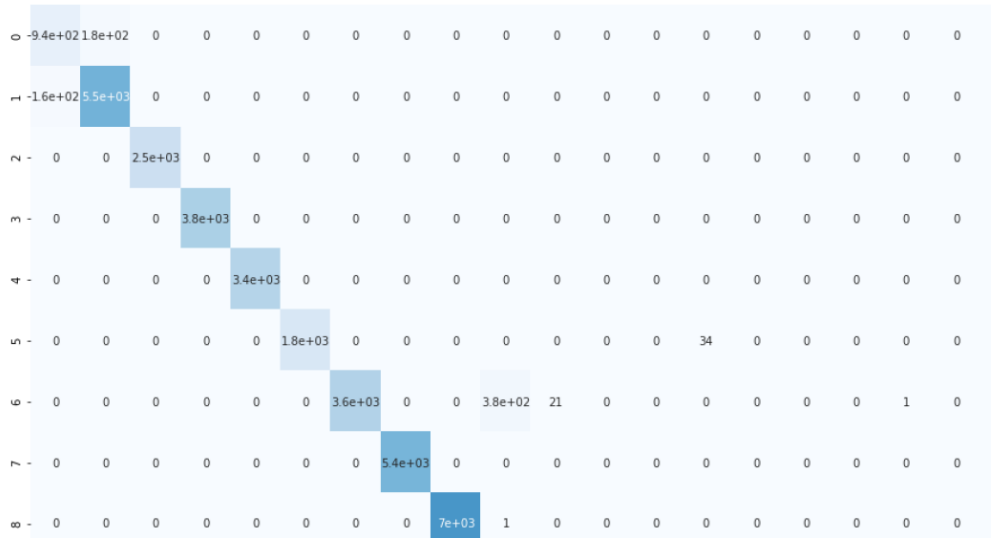

Figure 3.6: A small portion of the confusion matrix, extracted from a confusion matrix of large number of classes, demonstrates a strong diagonal component, indicating high confidence in the machine learning model's performance

where y represents actual residence time and $\hat{y}$ is the predicted residence time, a value closer to 1 is considered as a good prediction model. We see from Table 3.4 that the regressor performed very well in terms of median relative error (less than 2%) however with R^2 being low there was a relatively large variation of error around this median value, hence the need for the error margin added to the regressor predictor.

We trained our ML model on a single core server and it took approximately 18 hours for 4,500,000 transactions *i.e.* 70 ML operations per a second. The statistics of vehicular transitions indicate that median transition rate is one transition every 500 s for each vehicle; *i.e.* a single core can handle 35,000 simultaneous calls. Considering a typical European Nation like the UK may have approximately 1.5 million simultaneous calls[2] in a mobile network this means that less than 40 cores would be needed for a nationwide prediction service using this ML solution.

3.3.4 Call interruption results

After using the above to model the realistic call interruption we can determine how many calls have interruptions above a certain threshold as shown in Table 3.5.

[2]Estimated using data from Ofcom in the UK [114]

Table 3.5: Number of calls with interruptions greater than 300msec, 1sec, and 2sec respectively when considering the solutions with no prediction (NP), prediction of one network ID from previous network (1P), prediction of two network IDs from previous network (2P) and prediction with regressor and x second error margin (Px).

Scenario	300 ms	1 s	2 s
NP	**598,518**	**468,029**	**309,945**
1P	398,796	73,950	3,069
2P	**145,241**	**2,304**	**15**
P0	575,124	408,791	222,503
P2	563,410	352,343	135,022
P4	549,849	323,300	10,0745
P8	530,794	277,188	63,414
P12	513,637	235,098	39,733
P16	497,017	200,523	25,801
P32	444,940	118,136	7,647

Unlike packet loss in VoIP, there is little evidence to support what is an acceptable call interruption time. We have chosen three values from a level that is likely to be just perceptible in many conversations (300 ms, comparable to average word length) to a level of 2 s where it is a considerable portion of a typical sentence. The results in Table 3.5 indicate that call interruptions in the case of the longer interruptions of 2 s can be reduced by more than 99.9%, by using the case where two network attachment points are predicted (2P). Further detail on the call interruption times can be seen in Figures 3.7 and 3.8 showing that the prediction of two target network attachments (2P) gives substantially lower call interruption. It can further be seen that using the regressor for predicting the transition without an error margin (P0) does not achieve satisfactory reduction but that adding 32 s (P32) approaches the performance of the 1P case.

One of the costs of the mechanism proposed in this work is that there is additional traffic in the network during the bicasting. For the cases without the regressor (1P, 2P) this is when a mobile node moves to a new node. The overhead is shown in Figure 3.9 using the existing SIP handover without prediction (NP) as the baseline. For the case where bicasting is to one node (1P) or two nodes (2P), there is up to 2 times (1P) and 3 times (2P) the amount of traffic. The reason for double and thrice the amount of traffic is due to upfront media transmission on these predicted network attachment points during the handover. The overall distribution of handovers across the network attachment

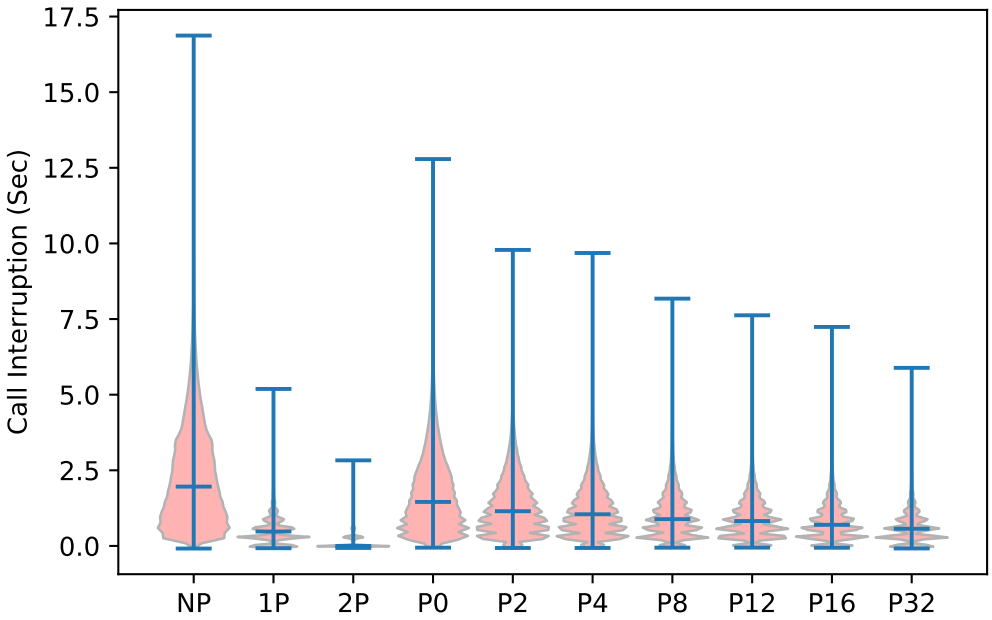

Figure 3.7: Call interruptions for the cases of no prediction (NP), using classifier to predict one (1P) or two (2P) network IDs, and using the regressor to predict residence time (Px) with varying regressor error margins x

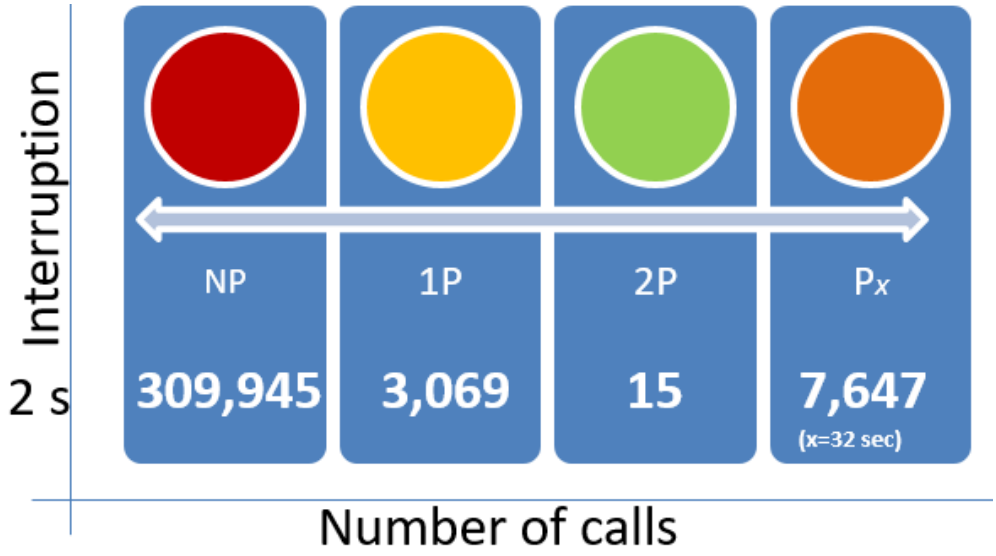

Figure 3.8: Interruption stats for calls with 2s interruption time

points is as shown in Figure 3.10 and the eCDF of network associated duration is shown in Figure 3.11.

The transition through the network attachment points is also not uniform as shown in Figure 3.12 showing that for some network locations, mobile nodes are attached to that network for a much longer time than other network attachment points; this is caused

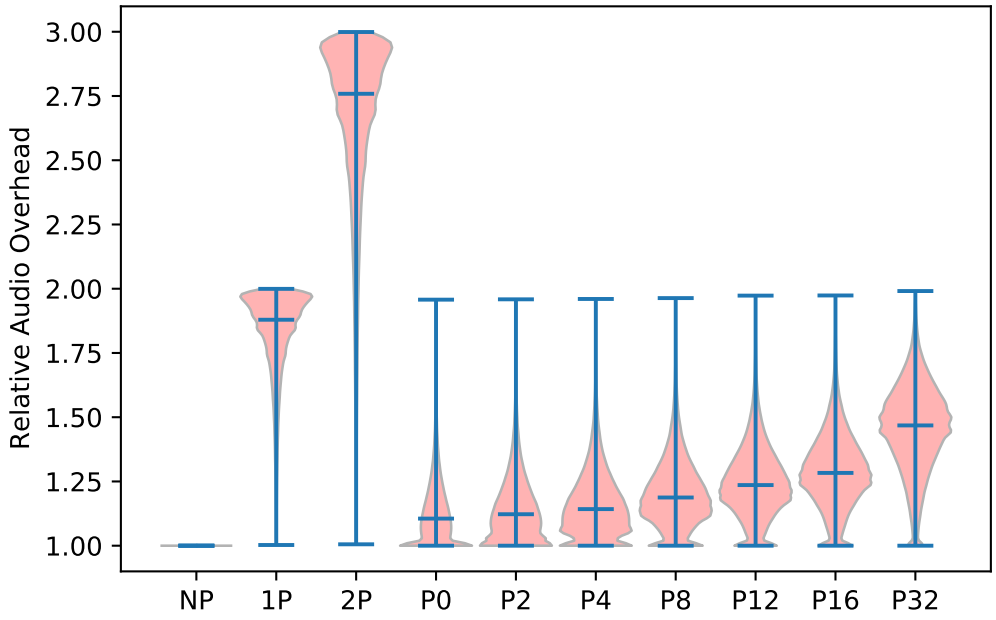

Figure 3.9: Relative Audio overhead for the cases of no prediction (NP), using classifier to predict one (1P) or two (2P) network IDs, and using the regressor to predict residence time (Px) with varying regressor error margins x

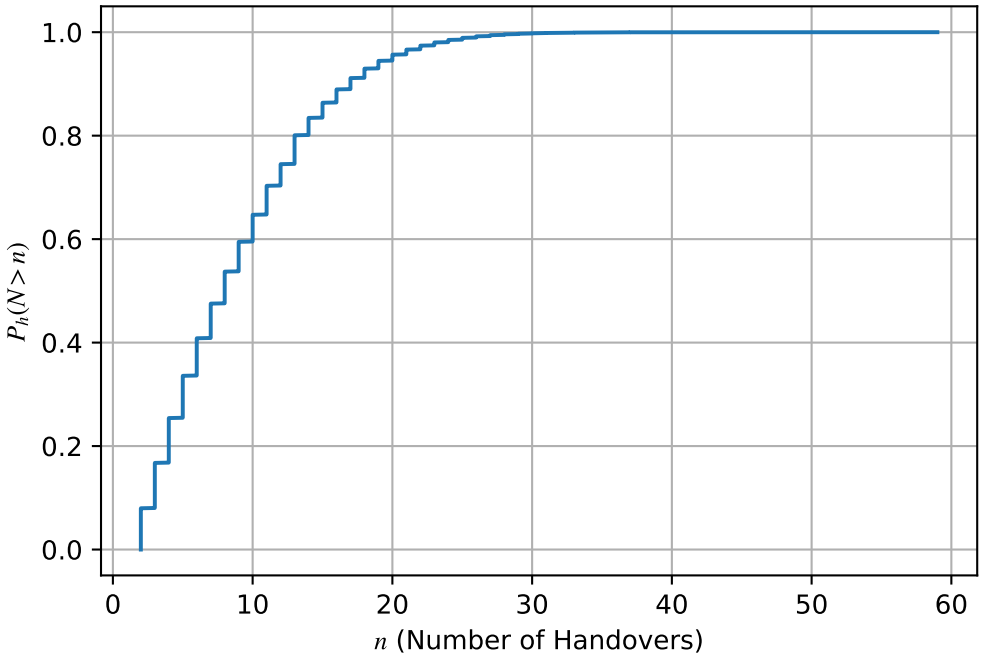

Figure 3.10: Empirical cumulative distribution showing number of handovers n for a call

by the diverse nature of network density across the geographical area with high density in the city centre and many transitions and much larger areas in the outer city zones with fewer transitions.

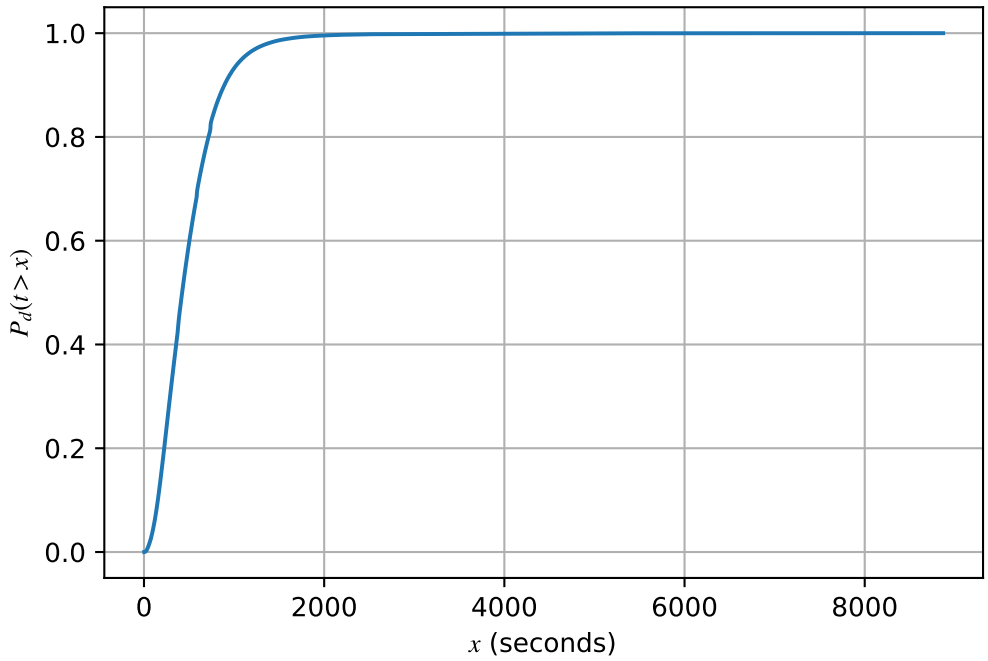

Figure 3.11: Empirical cumulative distribution function showing duration of network association t, median network association duration is 423s

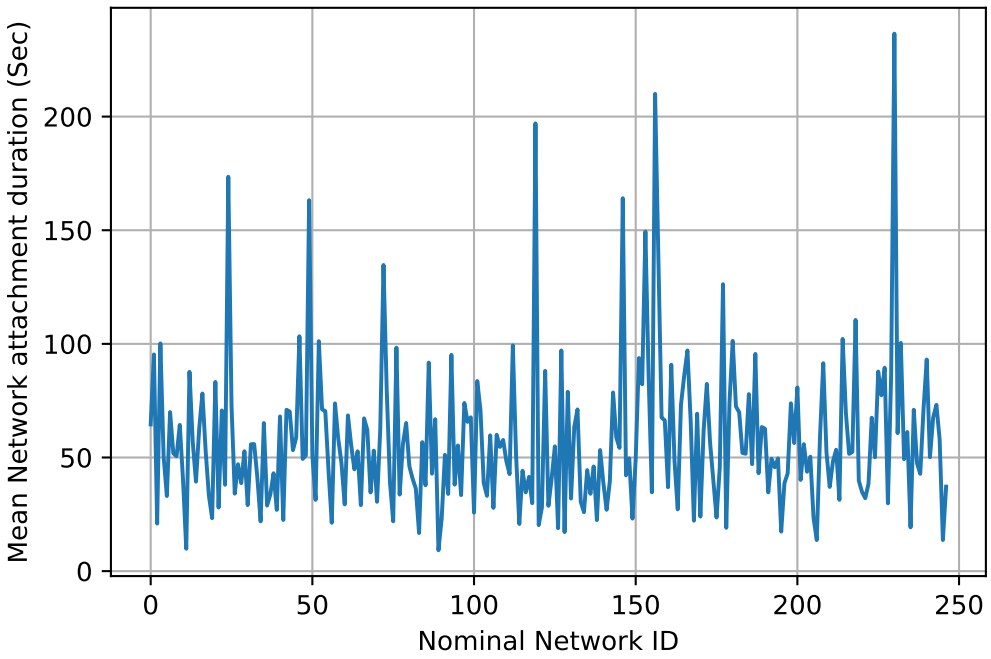

Figure 3.12: Mean network duration across the network IDs

3.3.5 Practical considerations

In practice a deployer of this solution would need to trade off the overhead shown in Figure 3.9 and the considerable benefit shown in Figure 3.13 and Table 3.5. Some

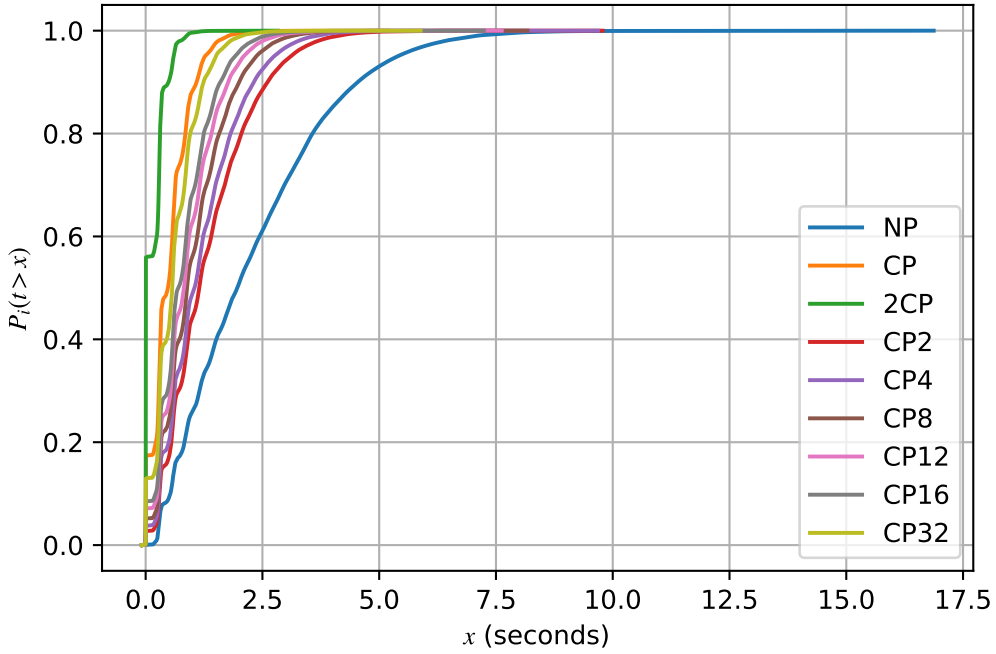

Figure 3.13: Empirical cumulative distribution showing call interruptions t

further insight is given in Figure 3.13 which shows the probability of call interruptions for the various options. It can be seen that there is a substantial benefit to deploying the ML solution in this work. While the exact trade-off would be a deployment decision, the author suggests that using the classifier with the regressor, using a suitable error margin gives optimal performance. Indeed using the regressor with 32 s error margin (P32) gives performance close to using the classifier alone (1P) but with substantially less transmission overhead by selectively bicasting. It should be noted that the 32 s suggested is significantly less than the median residence time of 423 s, as shown in Figure 3.11. It should be noted that the model must be trained with the respective local network transition dataset, incorporating up to 24 hours of transitions, when deploying this solution. In practice this can be obtained from the SIP Location Services [115], [116].

3.4 Conclusions

In this work, we have proposed a machine learning approach to avoid handover interruptions during the session updates in the application layer for pervasive communication

applications; this addresses the research gap and question on avoiding the SIP session latency during network handovers i.e. vertical handover as highlighted in Sections 1.3 and 2.8. A proactive handover *"make before move"* with bicasting of media sessions on two network attachment points; actual and predicted is proposed. Due to the unavailability of pervasive communication application transition data, a readily available user vehicular transition dataset is used for the handover analysis. This work compared several supervised machine learning techniques and the experimental results indicate that the random forests can perform well in these types of features due to the non-linear nature of the dataset. Mean call interruption is calculated on no prediction and prediction cases and results are plotted. The analysis shows that the more concerning call interruptions can be reduced by more than 99.9% with a modest communication transmission overhead. Our experiments conclude that the overhead of running the ML for handover prediction is small and that a single core running can serve the peak load of approximately 35,000 simultaneous pervasive communication sessions. Building on this work, it would be valuable to carry out subjective studies of the acceptable limits of temporal disturbance during handover. These limits could potentially be expressed in a similar manner to the impact of one-way delay in the ITU-T G.114 [117] specification.

Following the successful implementation of seamless vertical handover, the subsequent chapters concentrate on optimising horizontal device handover in an end environment that allows pervasive communication within that environment.

4

Seamless Device Horizontal Handover using MSC Predictor

Continuous growth in the smart speaker market has contributed to making high quality, far-field speech communications as a feasible alternative to the handset. Seamless handover offers a simple but effective way of improving the far-field communication experience by automatically switching to the best available device, in a single acoustic environment, regardless of where a user is located within that environment. This chapter proposes two significant solutions: reduction in media disruption during device handover by introducing a parallel session on multiple devices through session initiation protocol (SIP) call forking; and, coherence-based signal processing, MSC predictor, to more accurately determine the most suitable device for the user. The solution proposed uses the Magnitude Squared Coherence (MSC) and results verified through simulation and

real datasets show it has excellent performance. However, the raw MSC is found to have high variation due to room effects, consequently, this work shows that a smoothing predictor is needed to significantly reduce the extraneous transitions that would otherwise be subjectively poor. Unlike a purely location based approach, the proposed solution selects the best smart device without any environment specific calibration making it ideal for the straightforward deployment of a pervasive speech application that uses smart speakers.

4.1 Introduction

Chapter 1 outlines the primary objective of achieving a seamless vertical network and horizontal device handover in a pervasive speech system. The aim is to address the research questions and gaps identified in the pervasive speech system, to enhance the overall far-field communication experience by reducing handover latency and eliminating the need for manual call transfers across multiple devices. Chapter 2 explores the fundamental domain concepts, and Chapter 3 presents solutions to minimize vertical network handovers. The current chapter focuses on seamless horizontal device handovers, introducing, analysing, and providing a foundational solution for basic user movements. This foundational work is expanded upon in Chapter 5, where complex real-world scenarios are analysed. This chapter also proposes solutions for seamless session handover between the devices using the SIP *personal mobility* feature; this differs from the SIP solution in Chapter 3 as in this chapter we need to move the media within the same end-network *i.e.* doing horizontal handover. For more information on the SIP *personal mobility* feature, please refer the Section 2.2.1.1.

As a first step towards improving the horizontal handover, a basic seamless call handover experience has been demonstrated in the lab by BT to the author (work not yet externally described). The work by BT was carried out under the control of a call orchestration script with smart speaker development kits that have been used to provide basic user location information and an open-source call server used to perform call session handover between standard session initiation protocol (SIP) clients without any user intervention. While this demonstrated the technical merits of a pervasive speech system, a key requirement in moving from the lab to realistic environments is how well the user tracking works. While volume level and DoA (Direction of Arrival) angle have been sufficient at the Poof of Concept stage that BT investigated, the ability to work in realistic environments with no placement calibration is a significant challenge.

There are many ways to improve tracking reliability – such as making use of the addition of IoT sensors around the smart home – however, it is highly desirable to try

and derive this information from audio signals available from the devices that make up the communications system itself without requiring input from external devices or precise location calibration.

Consequently, this chapter addresses this need by using the *coherence* between audio signals in the smart device microphones. Instead of using precise location tracking, our work shows that correct transition between smart devices is possible using coherence measurement without any form of calibration. Additionally, this work considers how SIP can be used for mid-call session handover when using the audio features to identify the suitable device for handover.

The main contributions of this chapter are: proposing alternative approaches for SIP mid-call personal mobility; and, using the magnitude squared coherence, together with predictive smoothing, to automatically detect the audio device for session handover. This work has been published in a conference paper in 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA), Cairo, Egypt, 2022 [C2].

4.2 Proposed Method

In Section 3.2.1, the author has proposed alternative approaches *i.e.* bicasting for SIP session handling in order to achieve seamless session vertical handover using SIP *terminal mobility* support. This chapter proposes alternative approaches for SIP session handling in order to achieve seamless session horizontal handover using SIP *personal mobility* support. The key difference between the signalling in Chapter 3 and this chapter is that in this chapter we need to have multiple end device addresses handled in a transparent manner for the corresponding node. This is enabled through the use of the B2BUA in this chapter as a termination point. This means that the mobility can take place within the local environment, under the control of the B2BUA, without the corresponding node being aware.

4.2.1 SIP signalling for device handover in pervasive speech

The standard SIP protocol accommodates both *personal mobility* and *terminal mobility*, whether it occurs during pre-call or even mid-call. However, these methods do not account for smooth media transitions. In the context of *personal mobility*, it is possible to have several devices registered to a single SIP address for session management. Initiating a call session on multiple devices sharing the same SIP address can be accomplished through the existing SIP functionality, which supports either parallel or sequential call forking. Call forking is the process of processing multiple requests by B2BUA from/to the devices. In sequential call forking, the initial *Invite* is sent to a device and if the device is not answered, then new initial *Invite* is sent to the next device. This process continues until a device answers the call or all the devices registered under the same SIP address are exhausted. In the case of parallel call forking, initial invites are sent to all registered devices at once, and if one of the devices answers, then invites to other devices are canceled by sending the Cancel requests. The type of call forking to use is solely based on the requirement. This call forking capability can be leveraged for dynamic session handover between the smart devices. While a call is in progress, *i.e.* during a mid-call scenario, and a more suitable device is identified for handover, based on the method proposed in Section 4.2.2, the session handover can be achieved using the below methods:

Using existing SIP signalling: The existing session can be transferred to the new device using SIP Re-Invite or REFER methods. The message sequence flow at a higher level for this situation, involving call *sequential forking* and using REFER methods, is illustrated in Figure 4.1. Further detailed description of session handover flow is available in Appendix A.2.1.

Using modified SIP signalling: This thesis also introduces an alternative approach for session handover, as depicted in Figure 4.2. Slight changes are required to the existing B2BUA behaviour. However, this novel method eliminates media interruptions and offers improved control over session management for devices within the handover domain. A detailed description of session handover flow using this approach is available

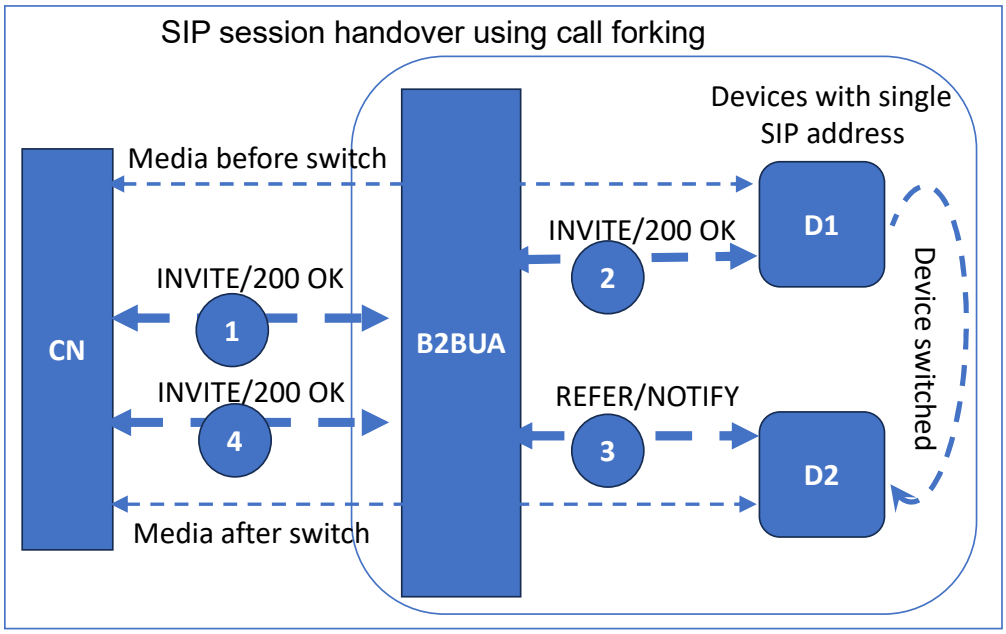

Figure 4.1: SIP mid-call horizontal device session handover using REFER/NOTIFY between back-to-back user agent (B2BUA) and corresponding node (CN)

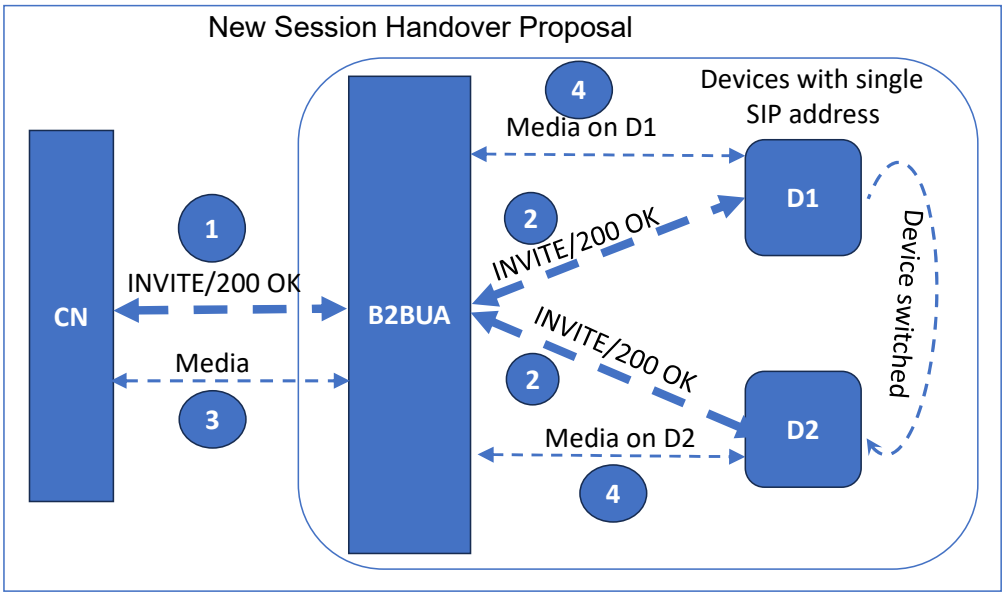

Figure 4.2: SIP mid-call horizontal device session handover with modified back-to-back user agent (B2BUA) and corresponding node (CN)

in Appendix A.2.2.

In practice media monitoring and switching is needed at a node in the network and it makes sense to carry this out on the B2BUA with the proposed modified SIP signalling which is straightforward to implement. In a practical deployment, this could be carried

out at the home/office gateway or in one of the smart devices, both of which are being used as a B2BUA in emerging systems.

4.2.1.1 SIP message sequence for the improved handover proposal

The proposed signalling solution is a two step process to enable SIP *personal mobility* i.e. multi device use in horizontal handover. The first step involves registering the devices to one SIP address. The sequence of the message flow for registering the devices to a SIP Registration is as described in Appendix A.1.0.1. Updating the SIP registry is described in Appendix A.1.0.2. The second step is the session initiation on multiple devices using initial *Invite* as described in Appendix A.1.0.3. The message flow for the session describes the sequence of the message flow from/to the B2BUA to/from all the registered devices in the new proposed solution. The flow uses the concept *parallel call forking* and the main changes are: after one device sends 180 Ringing, the other devices also sends 180 Ringing, and subsequent 200 OK and ACK messages are flown between B2BUA and to all the devices as shown in Figure A.4. It should be noted that the author is intimately familiar with SIP messages and the message flow is derived based on that knowledge. However, this SIP session handling has not been actually implemented.

4.2.2 Nearest device estimation

Here we propose how to perform the intelligent switching between devices within a single audio environment, for example, two smart devices (smart speakers) in a room or meeting room. The scenario assumes that a voice call is in progress and that the user wants to automatically swap between smart devices in the audio environment. We will consider only one end of the communication here, although of course the system could be used at both ends of the communication. As described in Section 4.2.1, the signalling and detection only take place at one end of the communication so that the other end (or ends in multi-party cases) is unaware. Additionally, we assume that the smart devices have at least two microphones; in practice smart devices tend to have two or more microphones.

The basic process involves capturing the audio from the smart devices and using the talker's voice to determine which device is best to use for continuing an audio call. While other techniques [118] propose using location identification, which usually involves precise calibration and device placement, here we aim to perform the detection using only the talker's voice without knowledge of the placement or orientation of the devices. We assume that the devices are equipped with two microphones, as is usual with most smart devices for echo suppression. We also assume that the talker's movement is relatively slow compared to the distances between the smart devices, for example walking from one end of a room to another (*i.e.,* not running between two closely spaced speakers).

As described in Section 2.5.1 the MSC has been studied in the past for localisation applications [90], [91], [94] and investigations have concluded that it is one of the best features for this purpose. Consequently, it was selected as the best method for analysing the audio in this chapter. Chapter 5 will consider some other features but broadly confirms that the MSC is the most useful. One clear justification for using the MSC is that it measures the coherence of the acoustic signals. When a talker is close to a device, the reflections from different surfaces (walls, ceiling, floor, furniture, *etc.*) tend to be much lower than the direct signal and thus the coherence between the signals received at two microphones in a device tends to be higher. Instead, when a talker is further away the multi-path signals from surface reflections tend to affect the speech signals at each microphone to a greater amount and consequently give rise to differences, lowering the coherence.

An overview of the proposed procedure is:

- Capture speech separately from the microphone pair from each device (Device 1 and Device 2 in examples later)
- Apply A-weighting to each audio signal to emphasise the key frequency components of speech
- Calculate the MSC, ρ_1, ρ_2, on a microphone pair from each device over an audio block (see the detailed explanation below)

- Apply a predictor to the output of the MSC over time to remove the room effects and low-level transient features due to room acoustics, obtaining the estimates $\hat{\rho_1}$, $\hat{\rho_2}$
- Determine the best device using $\max(\hat{\rho_1}, \hat{\rho_2})$
- Switch the call media to the best device using the signalling described in Section 4.2.1.

The above steps assume that the process is only taking place while the local user is talking. In practice, voice activity detection is needed to determine when this phase of the conversation is taking place, this is not considered in this work as this type of detection has been widely considered in the literature [119]. We now formally state the procedures.

4.2.2.1 Calculating the Magnitude Squared Coherence

The MSC is calculated using Welch's cross-power spectral density [120] for the two microphone signals from device i, $x_{i,l}(t)$ $x_{i,r}(t)$, for the left, l, and right, r, channels respectively. Specifically the MSC $\rho_i(t)$ for a block at time t is calculated using:

$$\rho_i(t) = \frac{\left|\hat{P}(X_l(f,t), X_r(f,t))\right|^2}{\hat{P}(X_l(f,t), X_l(f,t))\hat{P}(X_r(f,t), X_r(f,t))} \tag{4.1}$$

where the cross power spectral density $\hat{P}(X_a(f,t), X_b(f,t))$ is calculated across an N block Fourier transform of $X_a(f,t)$ from $x(t)$ as:

$$\hat{P}(X_a(f,t), X_b(f,t)) = \frac{1}{N}\sum_{f=0}^{N-1} |X_a^*(f,t)X_b(f,t)| \tag{4.2}$$

which denotes $X^*(f,t)$ as the complex conjugate of $X(f,t)$.

Finally the averaged MSC, $\hat{\rho}(t)$ at time across B blocks of size N with sample rate, r is

$$\hat{\rho}(t) = \frac{1}{B}\sum_{k=0,\tau=t+kN/r}^{k=B-1} \rho_k(\tau) \tag{4.3}$$

Unfortunately, the MSC calculated using this method is highly dependent on reflections in the room, which cause frequency dependent constructive and destructive interference, as will be shown in the results below. Consequently, we need to predict the MSC $\hat{\rho}$ from a number of previous noisy observations $\rho_t, \rho_{t-1}, \ldots$.

4.2.2.2 MSC predictor

The MSC is predicted using the double exponential smoothing method [121], [122], a popular time-series estimator that has been shown to be useful in location tracking applications with considerably lower computation costs than other predictors such as a Kalman filter [123]. The author attempted using a number of other time-series prediction algorithms [124], [125] such as Autoregression (AR), Moving Average (MA), Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA - The ARIMA model is a combination of two methods, AR and MA models). But due to the non-stationary nature of the data, the exponential smoothing has produced good results compared to the stationary time-series prediction algorithms.

The double exponential smoothing algorithm maintains level and trend estimates at each period and prediction of $\hat{\rho}$ is calculated based on weights α and β using:

$$L_t = \alpha\rho_{t-1} + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{4.4}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{4.5}$$

$$\hat{\rho}_t = L_t + T_t \tag{4.6}$$

where L_t represents the *level* and T_t represents the *trend* at time t. The values of α and β will be dependent upon the sampling rate of the MSC, ρ, and typical variations due to movement. In practice, it has been found that a Monte-Carlo optimisation with a relatively coarse-grained search allows α and β to be obtained with values that work well.

Table 4.1: Simulation setup - Standard room

Parameter	Description
Room dimensions	7m X 5.5m X 2.4m
Device positions	Device1: 1.0m X 3.5m X 0.9m Device2: 5.5m X 3.5m X 0.9m
Device details	2 microphone arrays, left and right, separated by 0.1m
Source positions	110 positions starting from 1m to 6m each with 0.05m steps
RT60	0.6s, 0.8s, 1.0s, 1.5s, 1.8s
Audio sample rate	16 kHz
Audio block size	1024 samples

Table 4.2: Simulation setup - Conference room

Parameter	Description
Room dimensions	10m X 7m X 2.4m
Device positions	Device1: 2.0m X 4.0m X 0.9m Device2: 5.0m X 4.0m X 0.9m
Device details	2 microphone arrays, left and right, separated by 0.1m
Source positions	115 positions starting from 1m to 7m each with 0.05m steps
RT60	0.6s, 0.8s, 1.0s, 1.5s, 1.8s
Audio sample rate	16 kHz
Audio block size	1024 samples

4.3 Results

4.3.1 Simulation setup

Two rectangular rooms with characteristics listed in Tables 4.1 and 4.2 were simulated using the Pyroomacoustics Python package [16]. One room represents a standard living and the second room represents a conference room. Both are simulated without considering artifacts such as variable wall absorbance, windows, and furniture. While these additional artifacts would change the specific MSC measured at a single point, this work is more interested in the relative difference between MSC at the two devices rather than the specific MSC value. In the standard and conference rooms, the two devices, Device 1 (D1) and Device 2 (D2) are simulated each with two omnidirectional microphones and placed at opposite ends of the room with a distance of 4.5 meters between them in the

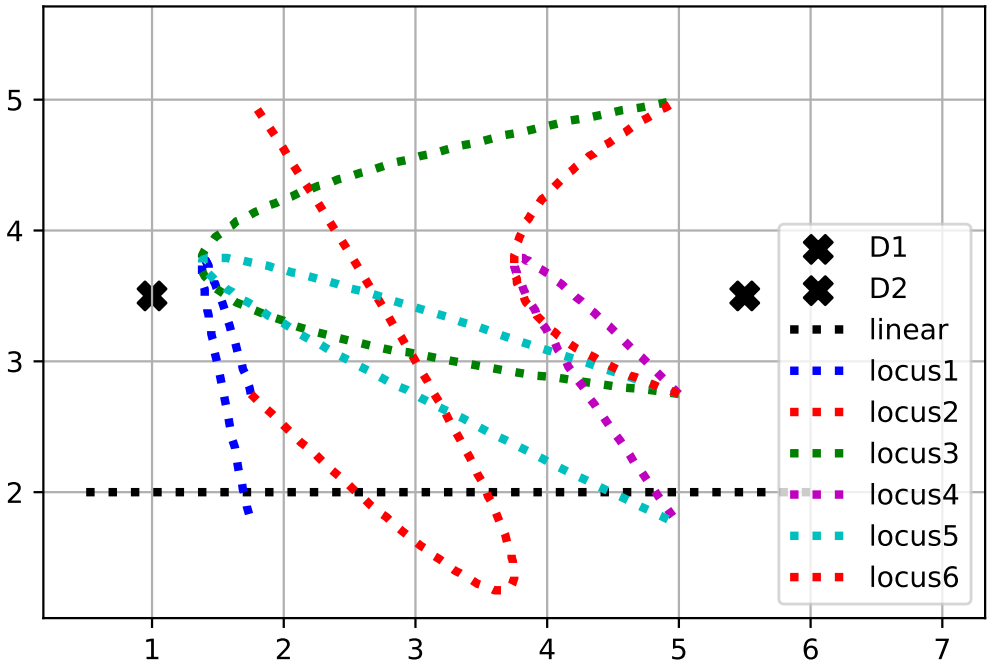

Figure 4.3: Linear and locus source positions compared to fixed device positions (D1, D2). A sample of six loci from the 400 generated are shown.

standard room and 3.0 meters distance in the case of conference room. The standard room simulates an environment such as an open-plan living space with say a device in the living room area and another in the kitchen area. The conference room simulates a large room with a big conference table and two devices placed at each corner of the table.

Movement of the sound source (the talker) within the environment is simulated using two methods, as shown in Figure 4.3: *linear movement* where the sound source is moved in 0.05 m steps from one end of the room to the other; and, *locus movement* where the sound source is moved in a pseudo-random manner using a set of three random positions (a minimum distance apart) that are connected using a cubic Bezier curve. For the locus movement, a set of 400 loci were created with each locus sampled at 100 equally spaced points along the curve. This simulates the talker moving with a purpose between two locations while navigating around a third object such as furniture. A representative set of six such loci are shown in Figure 4.3.

4.3.2 Device transition results

The MSC using Welch's method, along with applying A-weighting and double exponential smoothing for prediction, as described in Section 4.2.2.1, is captured for each position of the sound source on both devices. The device with higher MSC, $\max(\hat{\rho_1}, \hat{\rho_2})$, is considered as the best device to transfer the session. The experiments were performed in simulated rooms with various reverberation time as described by the RT60 metric *i.e.* moderate reverberation case (e.g. living room) to high reverberation conditions (e.g. concert room). The Monte-Carlo parameterisation determined values of $\alpha = 0.05$ and $\beta = 0.01$ as suitable for the double exponential smoothing.

The performance is evaluated using two metrics:

number of extraneous transitions: the number of additional device transitions using the predicted talker location compared to the actual nearest device

weighted normalised error (WNE): an error metric $\hat{E}$ normalised to the difference of the distance between the devices and the talker calculated across all the measurements (the set P):

$$\hat{E} = \sum_{i \in P} \frac{e_i \, |d_i|}{D} \tag{4.7}$$

where $\sum_{i \in P} \frac{|d_i|}{D}$ is a normalisation, $|d_i|$ is the absolute difference between the talker and Device 1 and the talker and Device 2; $D = |d_1, d_2|_2$ and e_i is a 0,1 variable which is zero if the predicted device is the same as the nearest device or one otherwise.

The use of extraneous transitions as a metric is important as users would find switching that is unnecessary as subjectively disturbing, the ideal is for this to be zero. The purpose of the WNE, $\hat{E}$, is to compute an error that is higher if the device chosen is significantly further away than the ideal device.

The results of the linear movement scenario compared to the ideal transition, and for various simulated room RT60 values in a standard room, is shown in Table 4.3. Additionally, examples of the output of the MSC output are shown in Figures 4.4, 4.5, 4.6 and 4.7 for RT60 of 0.6, 0.8, 1.0 and 1.5 respectively. These graphs show the ideal

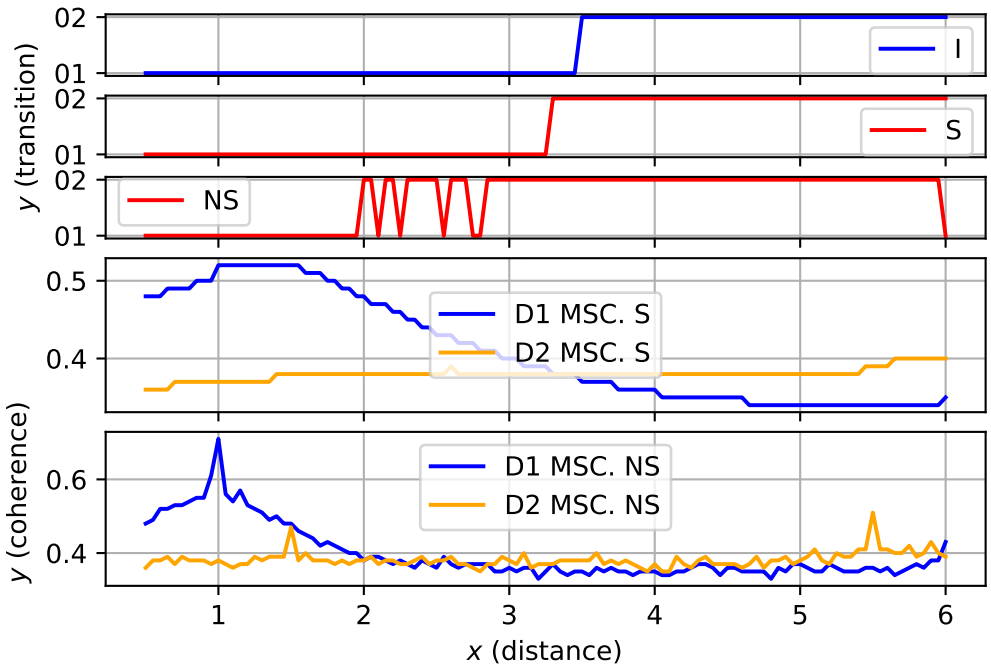

Figure 4.4: Device transition for Reverberation Time $RT60 = 0.6s$ showing ideal device transition (I), our, smoothed, solution (S), the non-smoothed case (NS) and the relative MSC values used to derive transitions.

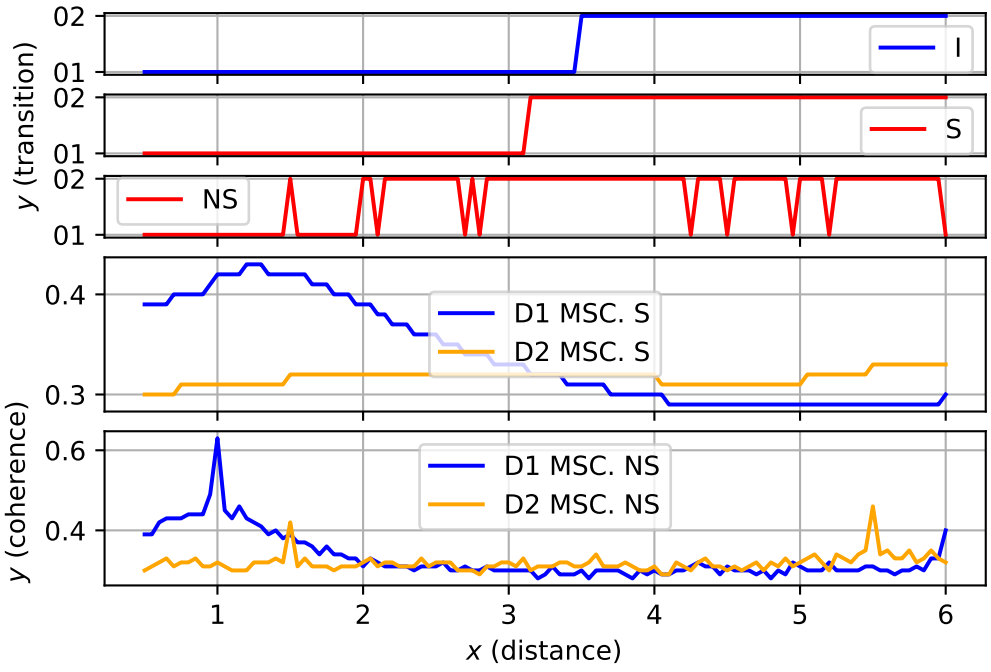

Figure 4.5: Device transition for Reverberation Time $RT60 = 0.8s$ showing ideal device transition (I), our, smoothed, solution (S), the non-smoothed case (NS), and the relative coherence values used to derive transitions.

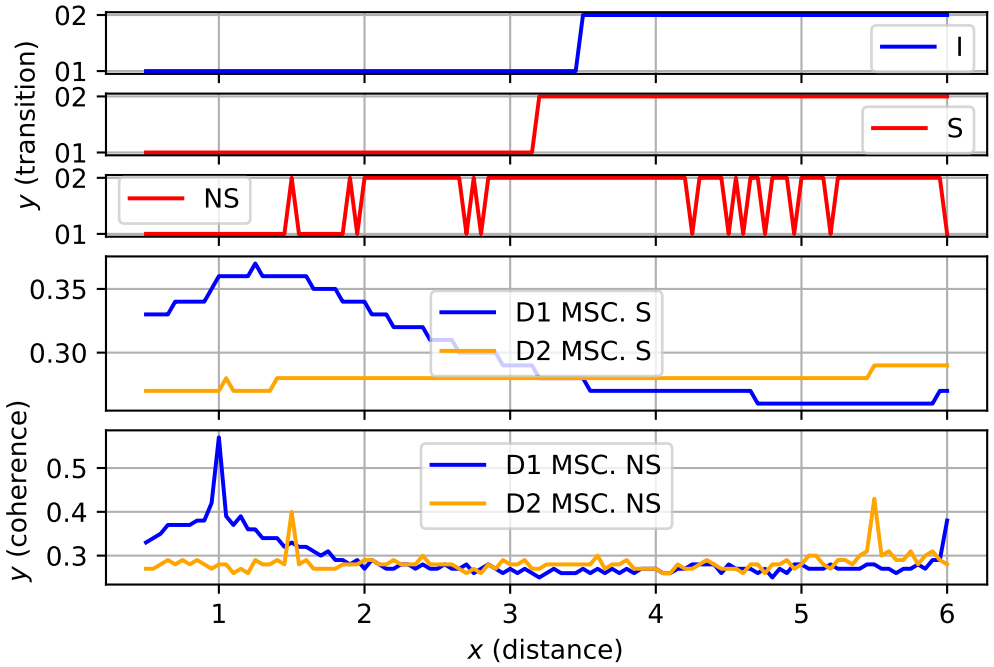

Figure 4.6: Device transition for Reverberation Time $RT60 = 1.0s$ showing ideal device transition (I), our, smoothed, solution (S), the non-smoothed case (NS), and the relative coherence values used to derive transitions.

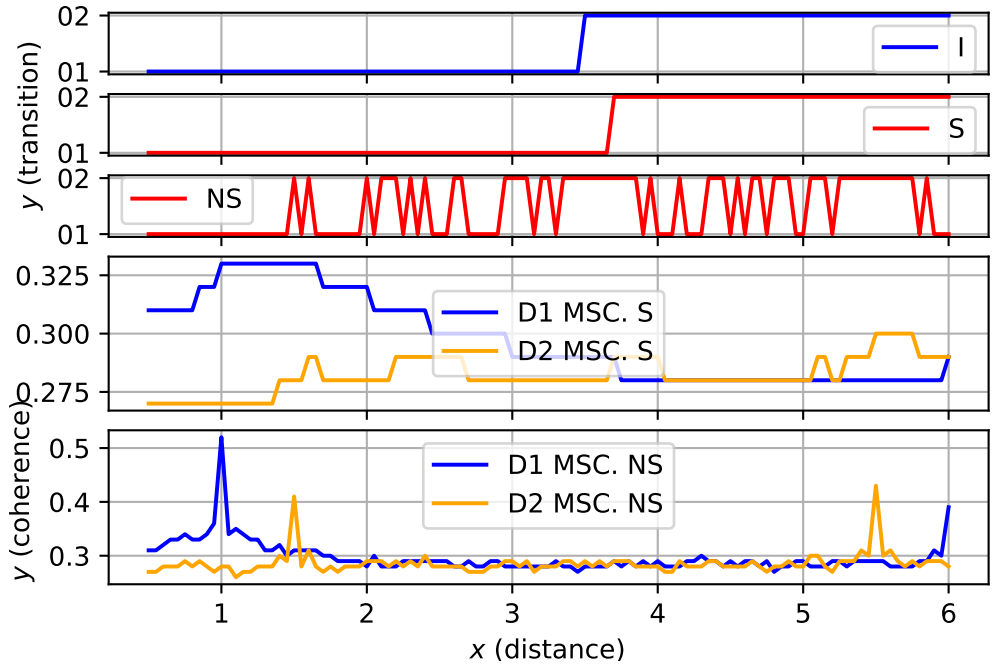

Figure 4.7: Device transition for Reverberation Time $RT60 = 1.5s$ showing ideal device transition (I), our, smoothed, solution (S), the non-smoothed case (NS), and the relative MSC values used to derive transitions.

Table 4.3: Comparison between different simulation scenarios using proposed metrics of weighted-normalised error (WNE) and number of extraneous transitions (ET) with MSC predictor

	RT60				
Type	0.6s	0.8s	1.0s	1.5s	1.8s
WNE-NS	0.14	0.20	0.24	0.25	0.3
WNE-S	0	0.01	0.01	0.01	0.18
ET-NS	9	17	22	38	38
ET-S	0	0	0	0	0

Table 4.4: Comparison between different time-series predictors against MSC predictor (Smoothed-MSC) using proposed metrics of weighted-normalised error (WNE) and number of extraneous transitions (ET)

Type	AR	MA	ARMA	ARIMA	**Smoothed-MSC**
WNE-NS	0.18	0.24	0.18	0.18	0.20
WNE-S	0.09	0.22	0.09	0.08	**0.01**
ET-NS	15	15	15	15	15
ET-S	7	35	5	3	**0**

(I) ground truth as the top blue line, then the results of using the proposed smoothed are shown in the top red line (S) and the non-smoothed are shown in the second red line (NS). Additionally, the MSC values ("coherence" on the graph) in Figures 4.4 and 4.7 show how the raw MSC values can vary significantly, with artifacts present due to additive reflections at certain positions; these are clearly worse with the higher RT60 value used for Figure 4.7. The NS (non-smoothed) data in the figures shows how the transitions cannot clearly be detected using the raw MSC, whereas the transitions in the S (smoothed) are very close to the ideal. It can be seen that the transition happens at not quite the optimal location if purely distance is used, however, in practice this is not likely to be subjectively important as users would want the device to switch appropriately rather than expecting millimeter precision on their location being exactly equidistant from the devices. Again the NS curve shows that, without the predictive smoothing, there would be many extraneous transitions that would be subjectively annoying. The results in Table 4.3 show that the proposed technique (MSC plus predictive smoothing) works very well compared to raw MSC results, which have an increasingly large number of extraneous transitions and error (WNE) as the reverberation increases.

The output of the MSC for the case of the conference room for RT60 of 0.6 and 0.8

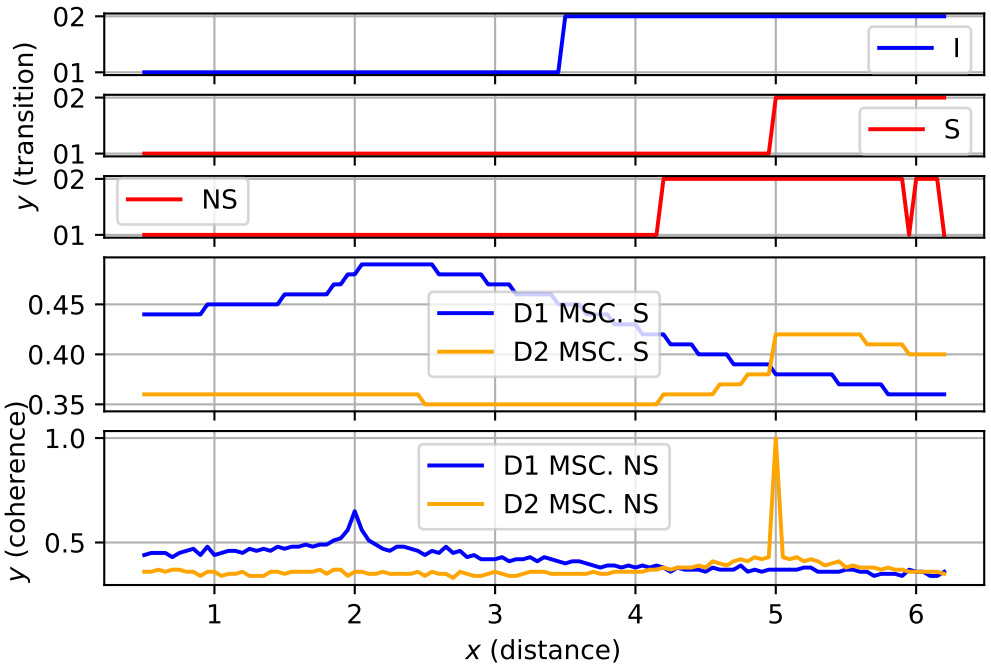

Figure 4.8: Device transition in Conference room for Reverberation Time $RT60 = 0.6s$ showing ideal device transition (I), our, smoothed, solution (S), the non-smoothed case (NS) and the relative MSC values used to derive transitions.

are shown in Figures 4.8 and 4.9 respectively. The prediction is performed with values of $\alpha = 0.05$ and $\beta = 0.01$.

We conducted a comparison between the MSC predictor employing double exponential smoothing for linear movement for RT60 of 0.8 and alternative time-series predictors, including AR, MR, ARMA, and ARIMA. The predicted MSC output and device transitions are illustrated in Figures 4.10, 4.11, 4.12, and 4.13. The outcomes of these figures and also the consolidated data presented in Table 4.4 depict the good performance of the MSC predictor when compared with other time-series predictors.

We also tested our work on more complex *locus* movement as described in Section 4.3.1 and examples shown in Figure 4.3. The results of the extraneous transitions for these scenarios is shown in Figure 4.14 showing that it was a more challenging problem; however, we can see that the number of extraneous transitions is considerably lower when using the MSC with predictive smoothing (S on the graph) compared to the raw MSC values (NS on the graph). We should note here that the same movement velocity was used in both the linear and locus movement, simulating a constant walking pace.

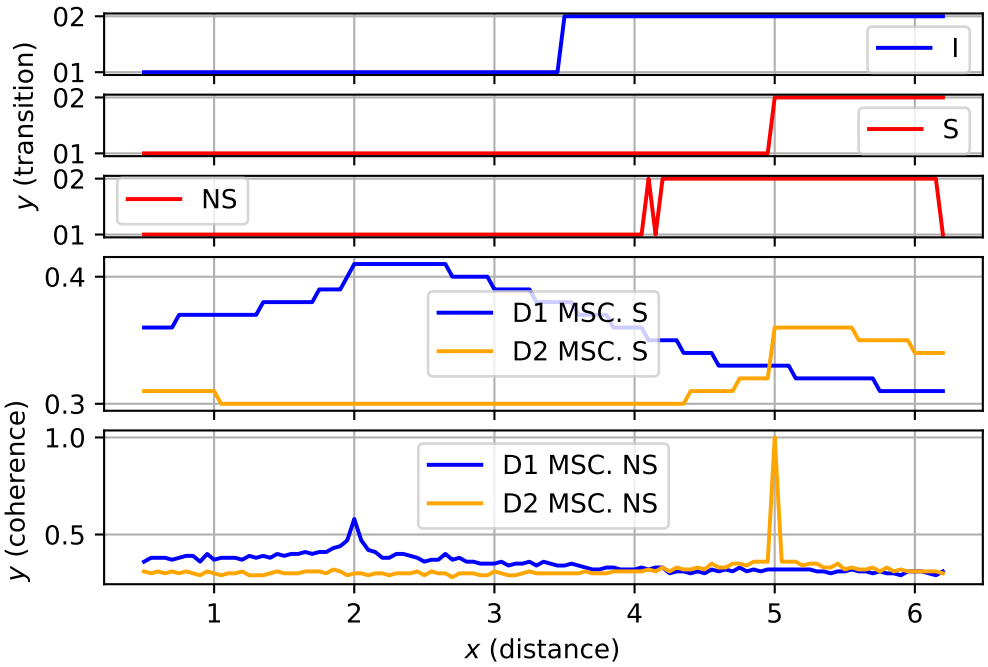

Figure 4.9: Device transition in Conference room for Reverberation Time $RT60 = 0.8s$ showing ideal device transition (I), our, smoothed, solution (S), the non-smoothed case (NS) and the relative MSC values used to derive transitions.

This limitation will be considered in Chapter 5.

Our solution was also verified on a real room/talker dataset obtained from [18], [19] where it achieved 92% accuracy in identifying the suitable device for a given talker location when static locations were used for the testing. This massive distributed array dataset was generated in the lab by Illinois Institute and contains two device types: wearable arrays and tabletop arrays. There are a total 12 tabletop arrays, each with 8 microphones, and 10 talker positions. Our solution *i.e.* calculating MSC, between the left and right hearing of microphone arrays, along with double exponential smoothing is applied on this dataset. The MSC was computed for each audio signal received from each talker across the eight microphone arrays, with each microphone participating in the comparison. The microphone with the highest MSC value was identified as the nearest speaker. The results concluded that our solution successfully identified the nearest device in all cases except one *i.e.* $11/12 \implies 92\%$ success. The single failure was for one device due to the reason of positioning of the talker facing away from the microphone arrays that were used to listen to the sound from the talker. One limitation

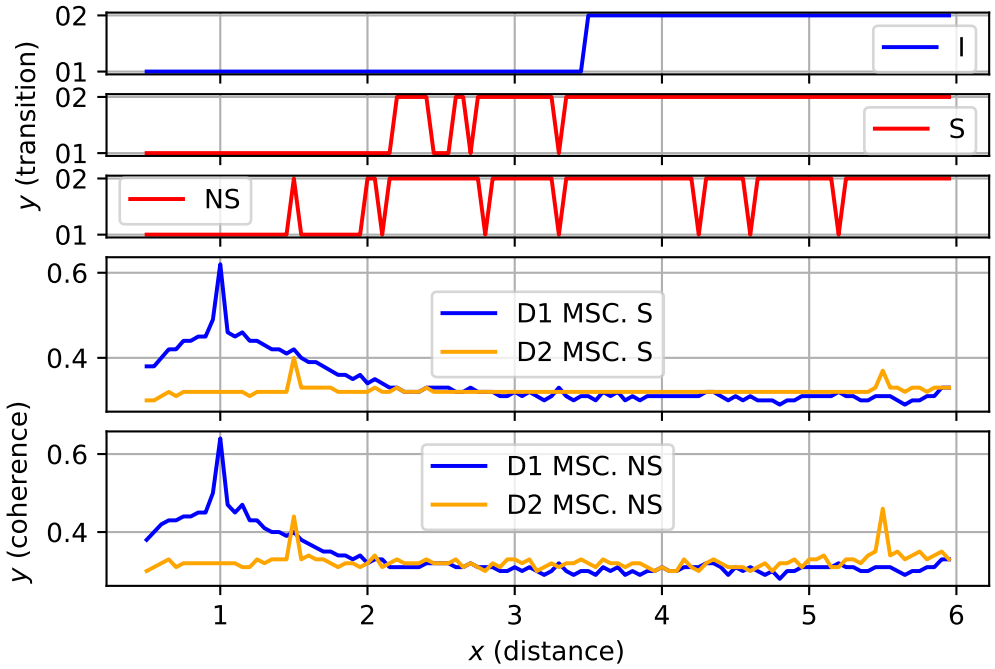

Figure 4.10: Extraneous transitions in Smooth (S) and in No smooth (NS) case using Autoregression (AR)

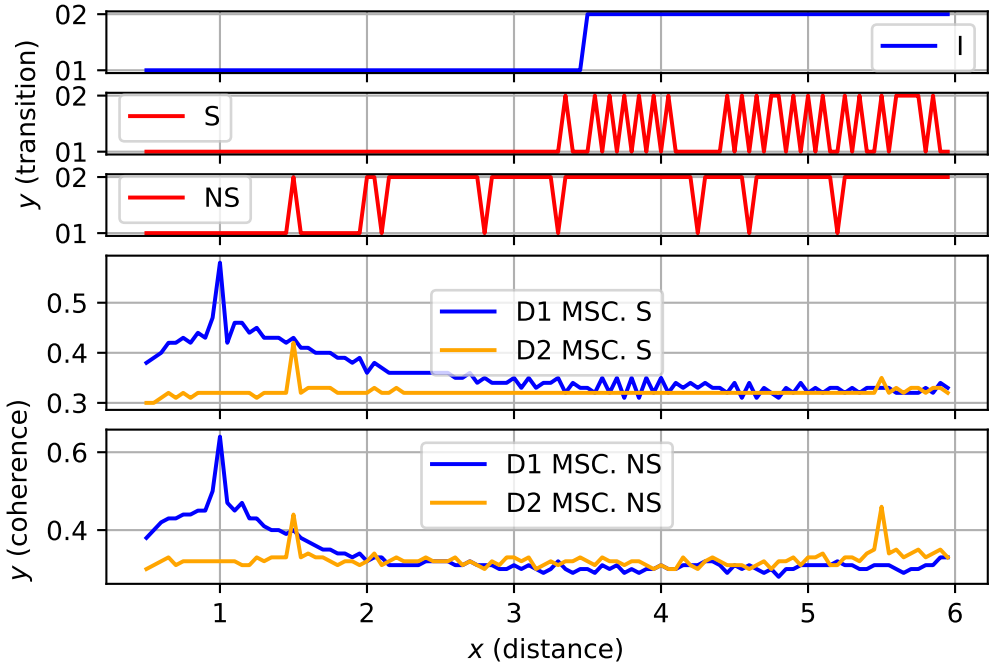

Figure 4.11: Extraneous transitions in Smooth (S) and in No smooth (NS) case using Moving Average (MA)

with this verification was that the real audio samples were captured at static locations, so while the results show that the method can discriminate correctly, it does not simulate real movement, again which will be better considered in Chapter 5.

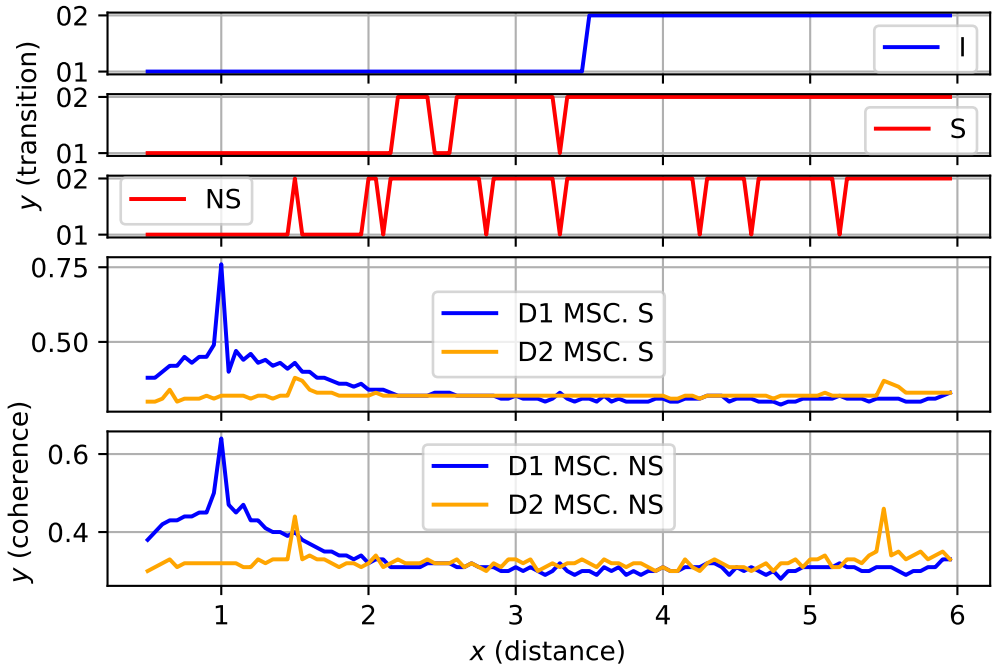

Figure 4.12: Extraneous transitions in Smooth (S) and in No smooth (NS) case using Autoregressive Moving Average (ARMA)

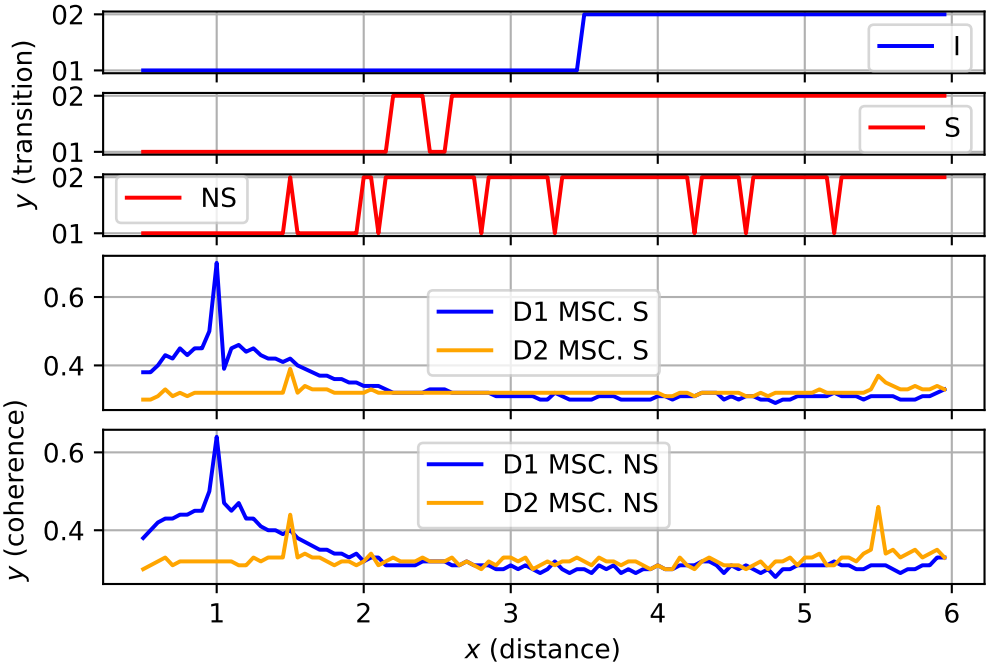

Figure 4.13: Extraneous transitions in Smooth (S) and in No smooth (NS) case using Autoregressive Integrated Moving Average (ARIMA)

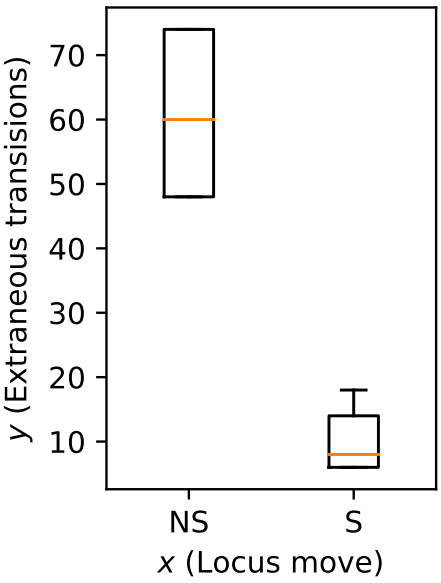

Figure 4.14: Extraneous transitions in Smooth (S) and in No smooth (NS) case for locus move

4.4 Discussion

The results show that the method proposed can work very well for detecting the appropriate device to switch a session to when two smart devices are part of a pervasive speech session. The method proposed does not use any actual location information or calibration of the device orientation. The metrics in the evaluation used a measure of error (the WNE $\hat{E}$) that determined a transition as an error if it did not occur at the equidistant point between the speakers, this was weighted by the "error" in the distance. However, in practice, it is unlikely that users will be concerned about the actual switching point being at the equidistant point, an error of a few tens of centimeters or even a meter or two may be perfectly adequate. Indeed consider a situation of an open-plan living environment with one smart device placed in a relatively acoustically dead living area and another placed in an acoustically live kitchen area. In this scenario, it may be beneficial for the switching to occur closer to the kitchen area so that the time spent using the device in the kitchen area is minimised to the user being located close to the smart device in this area. If the switching was done purely on distance (*i.e.,* actual location) then the non-ideal kitchen device might be used when it would be preferable

to use the device in the living area. The use of the MSC, as proposed in this work, conveniently obviates this problem as the MSC will tend to be less in acoustically live areas and higher in acoustically dead areas, such as the living room space of the example. The results above have only considered two devices. The work can be expanded to more than two devices by simply comparing the smoothed MSC returned by each device. To be precise if we have a set of D devices with the smoothed MSC of device $i \in 1 \ldots |D|$ represented as $\hat{\rho}_i$ then the selected device $\hat{d}_i \in D$ is determined using

$$\hat{d}_i = \arg\max_{d_i \in D} \hat{\rho}_i \tag{4.8}$$

4.5 Conclusions

This work proposes seamless mid-call session handover between the audio devices in a pervasive communication application using the SIP protocol and coherence based audio signal processing for detecting the correct smart device; this addresses the research gap and questions regarding seamless handover between devices within the same network, known as horizontal handover, as highlighted in the Sections 1.3 and 2.8. The analysis shows that MSC, along with a smoothing predictor, can provide a very accurate prediction of the optimum smart device for communication. The results show that a smoothing predictor greatly reduces extraneous transitions and transitions with greater accuracy compared to using the raw MSC values. The proposed method was verified in various room conditions for two simulated devices and also for a number of devices using a real room/talker dataset.

Deeper analysis, specifically the application of double exponential smoothing to a range of scenarios, including conversations in real-life settings at different speeds within the room, with varying signal strengths and with varying reverberation time (RT60), revealed unsatisfactory outcomes, as depicted in Figure 4.14. As a result, further research was conducted to explore the application of deep learning techniques to attain superior results. The subsequent chapter explores challenges encountered in a real-world acoustics setting, provides a comprehensive overview of the experiment methodology, and presents

the findings derived from this research.

5

Seamless Device Handover Through Deep Learning: a Custom Loss Function Approach

This chapter builds upon the foundational work established in the previous chapter, which introduced the concept of seamless device handover within a pervasive speech system to enhance the far-field communication experience. As a continuation of the overarching objective, this chapter delves into the difficulties of achieving seamless device handover for read-world scenarios, by exploring advanced techniques and solutions. The research extends beyond the initial lab-based proof of concept, addressing real-world challenges and refining the methodology for practical implementation. The ultimate goal is to transition from a foundational proof of concept to a refined, sophisticated, and effective solution capable of addressing the difficulties posed by real-world communication scenarios. Through a combination of data-driven approaches, further existing acoustic

simulations, and deep learning techniques, this chapter aims to overcome the limitations identified in the earlier stages of the research techniques.

5.1 Introduction

In Chapter 4, a fundamental coherence-based signal processing approach was introduced, employing a MSC predictor using double exponential smoothing to identify the suitable device for enhancing the far-field communication experience. However, when applying that solution to real-world scenarios involving talker movements at different speeds, various locus movements, varying signal strengths, and fluctuating reverberation time (RT60), the outcomes of the provided solution proved unsatisfactory. This chapter explores the advanced technologies to address real-world complexities. By leveraging data-driven insights and incorporating deep learning methodologies, we aim to optimise the device handover process thus setting the stage for a more sophisticated and effective solution. The proposed solution uses suitable device prediction based on a One-Dimensional Convolutional Neural Networks (1DCNN) using a custom loss function. Please refer to Chapter 2 for the core concepts of 1DCNN and details about its application in various domains. Within this chapter, our exploration starts with an in-depth analysis of the problem and emphasises the need for a custom loss function as an error function. We compare our 1DCNN with custom loss function results with MSC predictor solution, detailed in Chapter 4 and with the standard *Mean Squared Error (MSE)* as the error function. Through these comprehensive insights, we aim not only to address the shortcomings identified in earlier stages but also to establish a robust foundation for the subsequent implementation and validation of our advanced device handover methodology. This work has been submitted to the IEEE Transactions on Consumer Electronics, 2024 [J1].

5.2 Problem Statement

In Chapter 4 we introduced the problem in a simplistic manner. Here, the problem is strictly formalised so that it can be applied to both an ML model and used as part of the custom loss function that will be introduced later in the chapter.

Given N smart devices $D = d_1 \dots d_n$ in a room with a user at position $p(t)$ at time

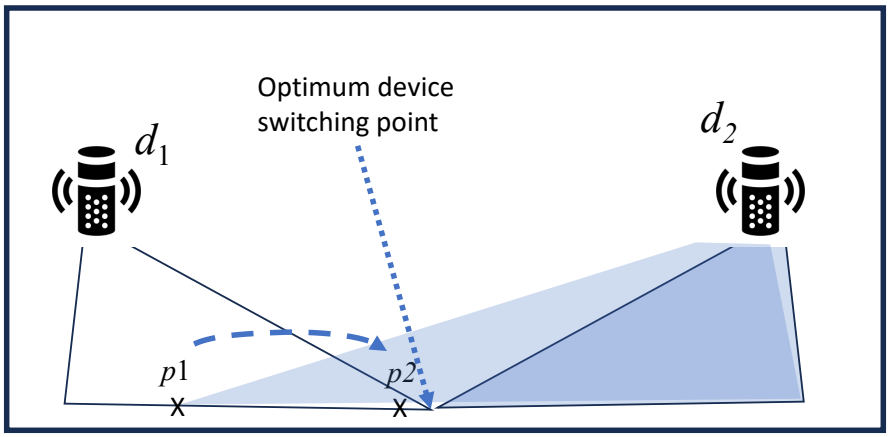

Figure 5.1: Diagram showing hypothetical switching error for two positions p_1 and p_2; switching to d_2 is perceptually much worse at p_1 than $p2$.

t using one of the smart devices to communicate with a remote user at a corresponding node (CN), the aim of this work is to determine an optimal $d_i \in D$ that will provide the highest quality communications experience. For simplicity, we assume that the user is moving and that the devices are static at locations denoted by the device variable d_i. On first inspection, this appears to be a simple classification problem. However, this simplistic approach leads to several issues. While there may be several ways to determine the optimum device, for the moment we will assume it is the device d_i that is nearest, according to Euclidean distance, to the user's position $p(t)$ and denoted as $|p(t), d_i|_2$. Later we will comment on this assumption. If we now consider the selection of the optimum device $\hat{d}_i$ as

$$\hat{d}_i = \arg\min_{d_i \in D} |p(t), d_i|_2 \tag{5.1}$$

the classification problem in this form thus depends upon determining the location of the user relative to the devices to obtain $|p(t), d_i|_2$. However, as noted above in Section 5.1 and in Section 2.6, location tracking in most environments where smart devices will be used (homes, offices) is very difficult; indeed given that users tend to place devices and move devices arbitrarily, even calibration based upon signals is very difficult. Consequently, determining $|p(t), d_i|_2$ is not practical and instead, we propose using features from captured audio on each of the devices to estimate the solution to (5.1). We describe the features we investigated for this work in Section 5.3.1.2.

In addition to the problem of location estimation, the classification formulation in (5.1) leads to issues when determining the error in a given position. A classical classification problem would evaluate "wrongness" as simple counts on the number of classification errors, but this misses some subtlety in the problem. Consider, for simplicity of description but without loss of generality, a two-device system $D = d_1, d_2$. Figure 5.1 shows the user in two positions p_1 and p_2. Again, assuming the shortest Euclidean distance as the decision variable for switching, it is clear that in positions p_1 and p_2 the speech should, ideally, be switched to d_1. Consider now if we use a binary error function, if in either case d_2 was selected we would have the same binary error value for both positions. However, this clearly does not match the requirements of the system as we see that p_2 is very close to the optimum switching point, thus an error in this case is much less perceptually important than an error at position p_1. In fact, if p_2 is sufficiently close to the switching point it will not be noticeable at all. Consequently, representing the system as a pure classification value does not adequately represent the problem. The next section presents our solution using an alternative representation of the error.

5.3 Proposed Method for Intelligent Device Handover

The intelligent handover of smart devices involves two key steps: firstly, the detection of the device, and secondly, the SIP session handover to the identified device. In this section, we elaborate on our 1DCNN technique, while deferring the discussion on session handover to the previous Chapter 4, as there are no alterations to the session handover process in this context. Section 5.3.1 will describe how the classification problem can be modelled as a *custom error* function, that will later be used as a *custom loss* function in machine learning. It then proposes the features that will be extracted for the 1DCNN and describes the structure of this model. After introducing the concept of nearest device detection in Section 5.3.1, detailed information regarding session handover can be found in Section 4.2.1.

5.3.1 Nearest device detection

A key challenge is to define a prediction function and an associated error function (later to be used as a loss function). To solve the problem found when using a binary classifier, as described above in Section 5.2, an alternative could be to consider it as a regression problem that determines the minimum distance to a speaker and uses this as a basis for switching. Now in our simple example, again in Figure 5.1, if $p2$ was incorrectly switched to d_2, the regression error would be much smaller than for p_1. However, while this regression problem has been widely used for systems that attempt to determine exact position, this is not practical in most smart device environments as we and others have found that predicting distance in arbitrary rooms with uncalibrated speaker positions and orientations is very difficult [91]. We choose to instead consider this as a hybrid classification/regression problem by using a normalised regression metric that is then fed to a simple decision classifier on the output of the normalised regression model. This hybrid solution is described as an error function that will be used as the custom loss function within the 1DCNN.

5.3.1.1 Custom error/loss function

The custom error/loss function is based on a combination of the normalised distance between the devices and a decision variable that indicates whether the ideal device was used. While Chapter 2 establishes the fundamental principles of the custom loss function, this section presents its application in the context of this research.

The computation of the custom error/loss function makes use of ground-truth variables that are known during training (and for evaluation), but are not known by the model during the prediction (running) phase. It should be pointed out that the custom loss function is only used during training. The proposed formulation of the custom loss function is:

$$L(t) = \left| \frac{|p(t), d_1|_2 - |p(t), d_2|_2}{|d_1, d_2|_2} \right| \delta(p(t), d_1, d_2) \tag{5.2}$$

where $\delta(p(t), d_1, d_2)$ is a binary decision variable which is unity if an incorrect device is

selected and zero otherwise and $|\cdot|_2$ signifies the Euclidean norm. All of the variables in (5.2) are unknown to a running system as we are assuming that the smart devices are placed (and possibly moved) by users without any calibration. Consequently, in the 1DCNN this loss function has to be implemented as a custom function such that the function (and input variables) are used during training but then hidden during the prediction (running) phase. While frameworks such as SciKit Learn [22] provide stub functions for custom loss functions, these are for generic use, and manipulating the system to include hidden variables requires some additional customisation.

5.3.1.2 Feature extraction

In this section, we present a method for intelligently switching between devices within a unified audio environment, using the audio features extracted at each talker's position and at a time interval. The process of extracting audio features follows a similar procedure outlined in Section 4.2.2 of Chapter 4, which presents the fundamental solution for device detection. However, notable distinctions exist; in this context, we account for user locus movements and extract a more extensive set of features beyond just the MSC feature. Rather than consider a continuous movement, the locus movement is broken up into small steps. In practice, this is needed as we will be working on blocks of audio, and also by averaging the features over these blocks, we reduce the computational complexity of the ML as it is working at the block size rate rather than the audio sample rate.

The methodology involves the use of D smart devices, such as smart speakers equipped with microphones, strategically placed according to user preferences within a room or meeting area. Here, the scenario assumes an ongoing voice call, and we presume that the user's movements are relatively gradual compared to the distances between the smart devices. For instance, the user does not traverse the room rapidly (running) but rather engages in expected movements during a call, such as walking. The process for computing the audio features is outlined below. Without loss of generality, we will consider the case where each device has a microphone array with two microphones as a *pair* denoted l, r. The processing system is thus described as:

- Capture an audio sample block, $(u_{1,l}, u_{1,r}), (u_{2,l}, u_{2,r}), \ldots$ separately from the microphone pair (l, r) from each device $d_i \in D$ at each time step
- Apply A-weighting filter A to each audio signal to emphasise the key frequency components of speech
- Apply a Hanning window, Han, on each block of audio, using overlap-add [126] to achieve continuity in the signal, $x = \text{Han}(A(u))$
- Calculate a relevant feature (below) from the audio signals $x_{i,l}$, $x_{i,r}$ for D devices, $i \in D$.

A number of features were considered in this work as specified below, these include the magnitude squared coherence, signal magnitude, Mel cepstral coefficients, and Mel magnitude spectrum. These were selected as they are commonly used, as highlighted in Section 2.5; each of these is formally defined below:

Magnitude Squared Coherence: The computation of MSC using Welch's cross-power spectral density is identical to the procedure described in Section 4.2.2.1 but is repeated here to maintain the context. MSC is computed for the two microphone signals from device $i \in D$, denoted as $x_{i,l}(t)$ and $x_{i,r}(t)$, representing the left (l) and right (r) channels, respectively. The MSC, indicated as $\rho_i(t)$ for a block at time t, is determined using the Fourier transform $\mathcal{F}$ of x, where $X = \mathcal{F}(x)$:

$$\rho_i(t) = \frac{\left|\hat{P}(X_{i,l}(f,t), X_{i,r}(f,t))\right|^2}{\hat{P}(X_{i,l}(f,t), X_{i,l}(f,t))\hat{P}(X_{i,r}(f,t), X_{i,r}(f,t))} \tag{5.3}$$

where the cross power spectral density $\hat{P}(X_a(f,t), X_b(f,t))$ is calculated across an N block Fourier transform of $X_a(f,t)$ from $x(t)$ as:

$$\hat{P}(X_a(f,t), X_b(f,t)) = \frac{1}{N}\sum_{f=0}^{N-1} |X_a^*(f,t)X_b(f,t)| \tag{5.4}$$

which denotes $X^*(f,t)$ as the complex conjugate of $X(f,t)$.

Finally the averaged MSC, $\hat{\rho}(t)$ at time across B blocks of size N with sample rate,

r is

$$\hat{\rho}(t) = \frac{1}{B} \sum_{k=0,\tau=t+kN/r}^{k=B-1} \rho_k(\tau) \tag{5.5}$$

Signal magnitude: The absolute signal magnitude (A), A_i, of device i is calculated from the power spectral density as:

$$A_i(t) = \frac{1}{N} \sum_{f=0}^{N-1} |X^*(f,t)X(f,t)| \tag{5.6}$$

where X^* is the complex conjugate of X.

Mel magnitude spectrum: The Mel magnitude spectrum $m(f_m, t)$:

$$m(f_m, t) = M(|X(f,t)|) \tag{5.7}$$

is a warped magnitude spectrum of $|X|$, the magnitude components of the discrete Fourier transform of x of a block at time t. Specifically, M is a filterbank that calculates the (mostly) logarithmically spaced frequency magnitude of the signal x at frequencies f_m. Here we used the spacing suggested by Slaney [127] which with a sampling frequency of 16 kHz gives the number of filters $H = 11$ in the filterbank M.

Mel Frequency Cepstral Coefficients: The set of Mel frequency cepstral coefficients (MFCCs) is an alternative *pseudo-time* domain representation of the Mel spaced magnitude spectrum. This is a common representation used in speech processing and is defined as a set of coefficients, $g_i(n)$, $n = 1 \ldots N-2$:

$$g_i(n) = \sqrt{\frac{2}{N}} \sum_{k=1}^{N-2} \log(m(k,n)) \cos\left(\frac{\pi k(2N+1)}{2N}\right) \tag{5.8}$$

which represents the discrete cosine transform of the logarithmically scaled Mel magnitude spectrum; note n=0, the DC component, and n=N-1 are ignored as they are usually zero magnitude components at these Mel frequencies in audio signals.

The author used Scipy [20] and Librosa [21] Python libraries to compute the audio features described above.

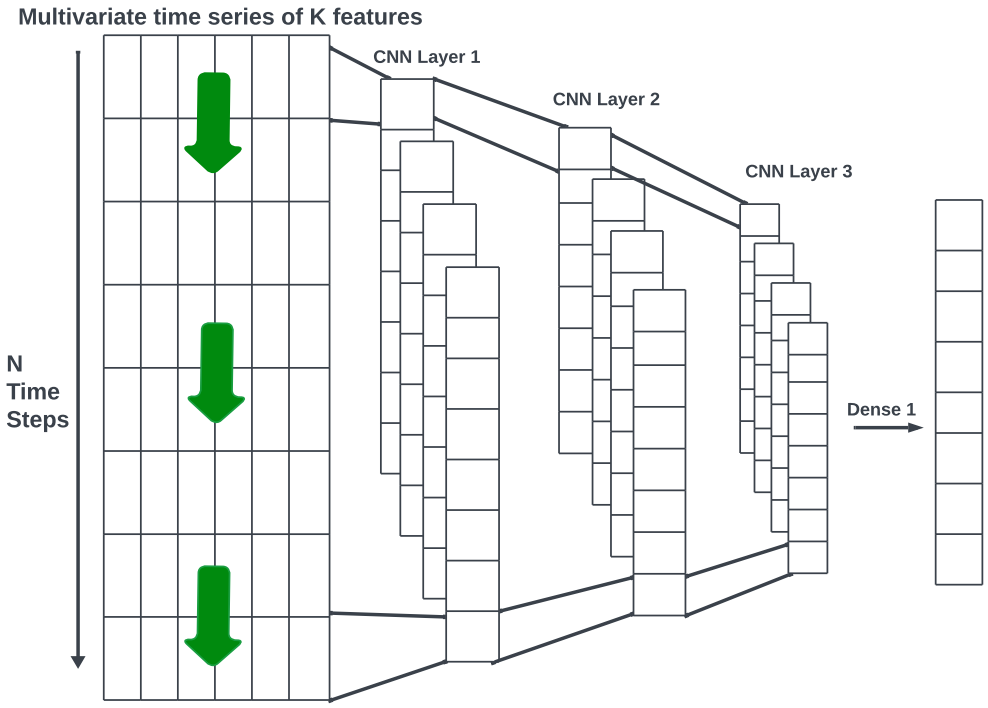

Figure 5.2: 1DCNN model

5.3.1.3 Machine learning architecture

This research proposes a 1DCNN to identify the most suitable smart device. The network processes time-series audio signal data using the multivariate features that were described in Section 5.3.1.2. The overview of our model is illustrated in Figure 5.2. In our model, we leverage multiple 1D convolutional layers to identify local patterns and features through the application of convolutional filters or kernels to the input data. Furthermore, we incorporate the Rectified Linear Unit (ReLU) activation function to infuse non-linear characteristics into the model. The ReLU activation function has the advantage of introducing some non-linearity while maintaining a linear function for a portion of the signal region. We performed a hyperparameter tuning process using the Random Search library from the Keras Tuner library, to identify the number of convolutional layers, filters and units to achieve optimal performance[1]. The specific hyper-parameters that were tuned include the number of filters and the kernel size. The tuning was carried out while using the custom loss function defined in (5.2).

[1]Hypertuning is a standard approach to optimising model parameters by automatically testing parameters over a reasonable range until good values are reached.

The input to the 1DCNN layer is a time-series, $\mathbf{z}(t)$, where each point in time is a vector representing the features described in Section 5.3.1.2. The input $\mathbf{z}(t)$ is convolved with a kernel $\mathbf{w}(t)$ of size l to obtain the output $\mathbf{C}(t)$, which is described using:

$$\mathbf{C}(t) = \mathbf{z}(t) * \mathbf{w}(t) = \sum_{k=-l}^{l} \mathbf{z}(k) \cdot \mathbf{w}(t-k) \tag{5.9}$$

The weights of the kernel $\mathbf{w}(t)$ are initialized using He normal initialization [128]. Then, the output of the CNN layer can be represented as:

$$C_i^l = b_i^l + \sum_k \mathbf{C}_k^{l-1} * \mathbf{w}_k^l \tag{5.10}$$

where C_i^l is the i^{th} output feature at the l^{th} layer, C_k^{l-1} is the k^{th} input feature at the $(l-1)^{th}$ layer, $\mathbf{w}_k$ denotes the convolution kernel at the k^{th} index, and b_i^l is the bias term for the i^{th} output feature at the l^{th} layer.

ReLU activation is applied on the convolution output:

$$ReLU(C_i^l) = \begin{cases} C_i^l & \text{if } C_i^l > 0 \\ 0 & \text{if } C_i^l <= 0 \end{cases} \tag{5.11}$$

The output of the final dense layer is a single regression value which is then mapped to a class through a binary decision variable, *i.e.*. the choice of device. The choice of the structure was driven by the observation that smoothing a single feature (e.g. MSC) can aid a very simple decision algorithm. Thus, similarly, the use of 1DCNN layers can be seen to act as time-domain variant trained filters that learn to appropriately process the input data to achieve improved classification performance. The size of the kernels is such that a reasonable history is used to feed into the classification decision (Section 5.4.2) for the values selected. We found, through experimentation and hypertuning, that it was best to maintain the time-domain structure throughout the CNN structure, with decreasing kernel size, until the end where the dense layer then reduced the dimensional to one.

5.3.1.4 Training environment

The training dataset comprises information such as the talker's position from the smart devices and the signal features, extracted from the audio outcome from each smart device microphone array, while the talker is in motion. The model will be trained using the talker's position as the ground truth and target variable, with the extracted audio signal features serving as the input features. Please refer to the Section 5.3.1.2 for further details on feature extraction. The ground truth labels for the optimum selected device used integers to represent the target device, in this case simply $[1, 2]$. To acquire these features, the system can undergo training via two primary methods: *real-world* data collection in actual rooms with people; or using *room simulations*. In the real-world data collection approach, training data would be collected by deploying smart devices in real rooms where actual people are tracked and interact with the devices. Conversely, in the room simulation method, the training data is collected by simulating a room layout furnished with multiple smart devices with microphone arrays, simulating various reverberation times to emulate authentic sound environments. The talker's movement is simulated by positioning simulated sound sources at different locations among the simulated devices, from which audio features are extracted. Simulated environments offer the advantage of accumulating data that encompasses various device setups, room configurations, and various talker locations. This research leverages a training dataset acquired from a simulated environment, and information about our simulation setup is available in Section 5.4.1.

5.3.2 SIP signalling for pervasive device handover

This work also considers SIP signalling and its *personal mobility* feature for the dynamic session handover to the detected device. During an ongoing call, *i.e.* in a mid-call scenario, if a more suitable device for handover is identified using the method outlined in Section 5.3.1, then the session handover can be executed using the methods described in Section 4.2.1.

Table 5.1: Room simulation setup

Parameter	Description
Room dimensions	7m X 5.5m X 2.4m
Room 1	Device1: 1.0m X 3.5m X 0.9m Device2: 5.5m X 3.5m X 0.9m
Room 2	Device1: 1.5m X 3.5m X 0.9m Device2: 6.0m X 3.5m X 0.9m
Device details	2 microphone arrays, left and right, separated by 0.1m
Source positions	200 locus movements each with 50 time steps from 0.5m to 6m
RT60	0.8s, 1.5s
Audio features	All, C, C+A
Number of audio signals	50
Audio sample rate	16 kHz
Audio block size	3200 samples (0.2 s)

5.4 Results

The evaluation was designed to test the proposed method against a non-machine learning approach and to test the proposed custom-loss function against a commonly used ML loss function. The evaluation is performed across 20,000 different scenarios: 50 different speech signals (taken from [17]) were applied across 200 different sets of movement loci and two different room scenarios (Room 1 and Room 2). The speech samples and loci in the training set were different from those used in the testing set. Room 1 was used for the training and then testing was applied in both Room 1 and Room 2. This section describes the simulation environment used for evaluating the proposed ML-based technique and a comparative method that employs a basic predictor solely based on MSC, without using calibration, as elaborated further in Chapter 4. An additional comparison is made by using a standard mean squared error as a loss function for the ML technique. First, we explain the room simulation environment and the 1DCNN model configuration. Then, before presenting the main results, we show a number of graphs that illustrate the operation of the comparative methods and our proposed 1DCNN based approach.

5.4.1 Room simulation setup

The process of room simulation follows a similar procedure as described in Chapter 4. However, notable distinctions for room simulation include variations in device and sound positions, the utilisation of different audio signals, and the extraction of a distinct num-

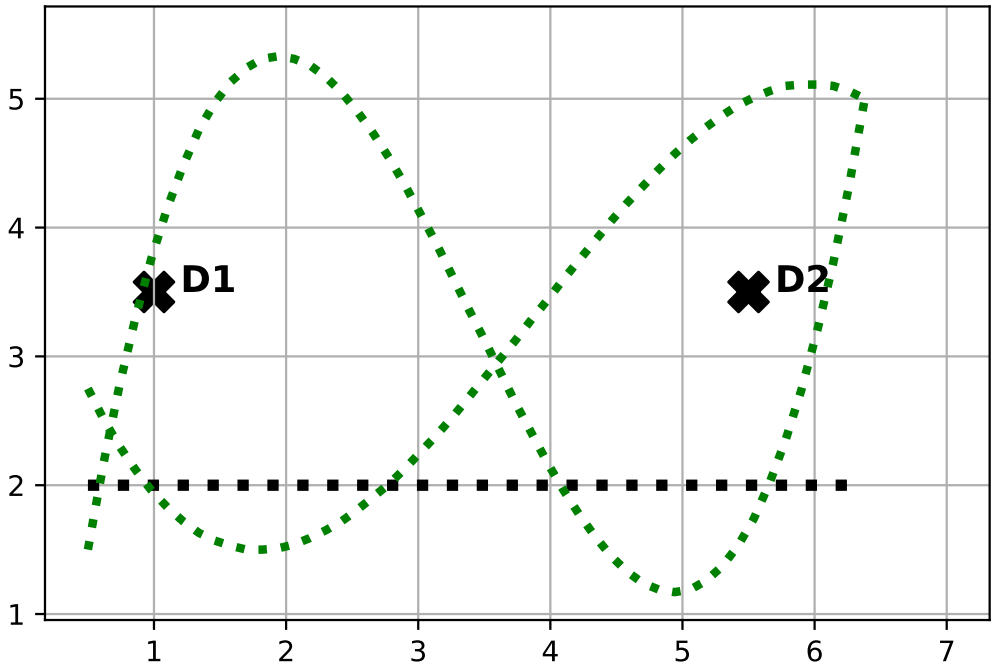

Figure 5.3: Two examples of simulated movement: black is a linear move, green is one of the 200 simulated random motions using smoothed Bézier curves. The dimensions of the room are in metres and examples of device locations are shown.

ber of audio features compared to the previous chapter. The procedure is described as below:

Two rectangular rooms, characterized as described in Table 5.1, were simulated using the Python package Pyroomacoustics [16]. These room scenarios were designed to emulate a typical living space, though certain factors like varying wall absorbance, windows, and furniture were not taken into account. As such, the experiments are conducted in an environment characterized by a substantial reverberation time (RT60) [103], replicating the conditions found in real-world room settings to mimic realistic sound environments. Within this typical room configuration, we simulate two devices, denoted as Device 1 (D1) and Device 2 (D2). Each device is equipped with two omnidirectional microphones strategically positioned at opposite ends of the room, maintaining a separation distance of 4.5 meters between them. This setup emulates a scenario resembling an open-plan living area, where one device resides in the living room section and the other in the kitchen area.

The movement of the sound source, represented by the talker's voice, within the

Table 5.2: Hypertuned 1DCNN model

Parameter	Description
1st layer	filters: 120 kernel size: 45
2nd layer	filters: 124 kernel size: 30
3rd layer	filters: 112 kernel size: 15
dense layer	1
activation	ReLU
strides	1
padding	same
number of time steps	49
audio features	All, C, C+A

Table 5.3: Genarated 1DCNN model with six audio features

Layer	Output shape	Param
1st layer	(None, 49, 120)	120 * (45*6 + 1) = 32520
2nd layer	(None, 49, 124)	124 * (30 * 120 + 1) = 446524
3rd layer	(None, 49, 112)	112 * (15 * 124 + 1) = 208432
dense layer	(None, 49, 1)	(112 +1) * 1 = 113

environment is simulated through two methods, as illustrated in Figure 5.3. The first method is a basic *linear movement*, where the sound source is incrementally moved in 0.05-meter steps from one end of the room to the other. The second method is *locus movement*, where the sound source follows a pseudo-random path defined by three distinct random positions, each separated by a minimum distance, and connected using a cubic Bezier curve. In the case of locus movement, a total of 200 loci were generated, with each locus being sampled at 50 evenly spaced points along the curve. This simulates the talker's purposeful movement between two locations, covering the two devices, while maneuvering around a third object, which could be furniture or any object in the room.

5.4.2 Configuration of hypertuned 1DCNN model

The high-level model is depicted in Figure 5.2 showing three convolutional layers that were determined through experimentation to be the best after experimentation in combination with hyperparameter tuning. The parameters determined after hypertuning are shown in Table 5.2. The input to the first layer comprises a number of audio features captured at 49 time steps. The number of different audio feature sets were tested in dif-

ferent combinations including MSC (abbreviated as just C); absolute signal magnitude (A); MSC with magnitude (C+A); and, a combination of the former with the addition of MFCC and Mel spectral amplitude (All). In the initial convolutional layer, we employ 120 filters, each responsible for learning distinct patterns or features from the input data, and these filters have a kernel size of 45. The configuration of subsequent layers was as determined by the result of hypertuning. Across all layers, a stride of 1 is used, meaning that the convolutional kernel moves one-time step at a time. Additionally, we apply the Rectified Linear Unit (ReLU) activation function to introduce non-linearity into the model, while ensuring also a linear portion of the transfer function. To maintain the output dimensions the same as the input, we are using "same" padding, which pads zeros if necessary on output from each layer. To produce a single predicted output, we conclude the model with a dense layer consisting of one unit. This dense layer takes inputs from the third layer in the model, computes a weighted sum of these inputs using a single set of weights, and generates a single predicted output. The generated model summary is depicted in Table 5.3.

5.4.3 Comparative techniques

As noted earlier we are interested to see how our approach compares against a simple MSC predictor and how the custom loss function improves the performance of the 1DCNN as described below.

5.4.3.1 Using MSC predictor

As a comparative method, we employ the MSC predictor, a solution proposed in the previous Chapter 4, specifically a double exponential smoothing predictor based on the MSC (Smoothed-MSC), as advocated by several other location tracking applications [129]. The primary benefit of employing this method is that it does not need any calibration for device detection. Using this technique, the device exhibiting the higher MSC, denoted as $\max(\hat{\rho_1}, \hat{\rho_2})$, is identified as the optimal choice for session transfer. The Monte-Carlo parameterization identified that values of $\alpha = 0.05$ and $\beta = 0.01$ are

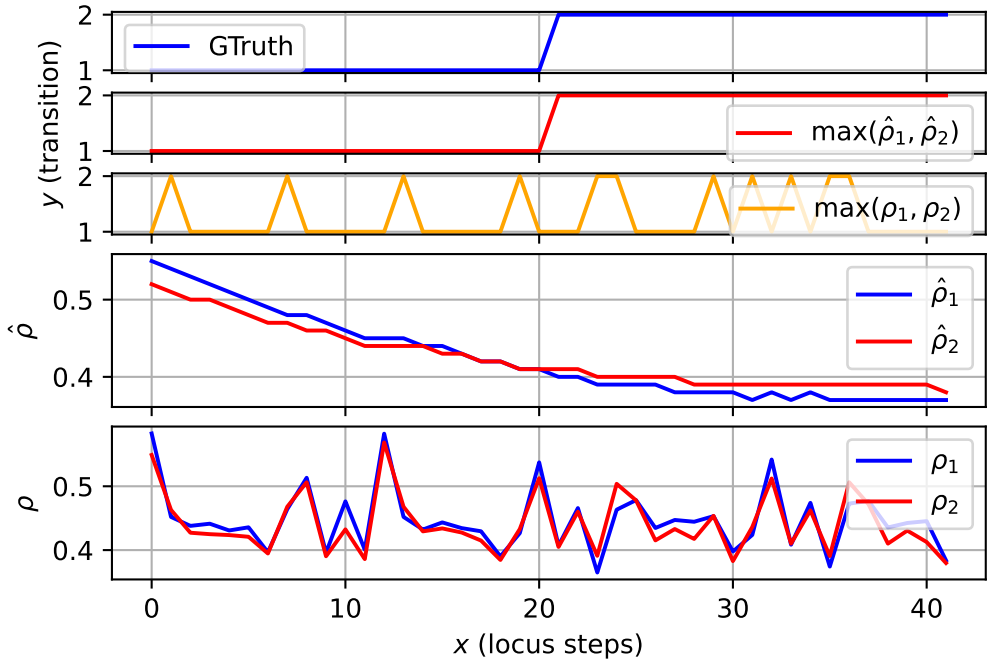

Figure 5.4: Device transition for a simple locus, showing ideal device transition (GTruth), MSC predictor solution (max($\hat{\rho}_1, \hat{\rho}_2$)), the non-smoothed case (max(ρ_1, ρ_2)) and the relative MSC values used to derive transitions.

appropriate for the double exponential smoothing applied to the raw MSC values. The details of the outcome of this technique is described in Section 5.4.4.

5.4.3.2 Using mean squared error as a loss function

As an additional comparison technique, we trained our 1DCNN model using a standard *mean squared error* (MSE) as a loss function. The outcomes of employing this technique clearly highlight the issue outlined in Section 5.2, emphasizing the need for a customized loss function as detailed in Section 5.3.1.1.

5.4.4 Device prediction results

As noted earlier, we first here present some graphs showing the operation of the Smoothed-MSC comparative method, shown in Figures 5.4, 5.5, 5.6, and the 1DCNN approach shown in Figure 5.7. The graphs show the ground truth (GTruth) as the top, blue, line as specified in (5.1) and the result of either the Smoothed-MSC (max($\hat{\rho}_1, \hat{\rho}_2$) - using MSC predictor) or the output of the 1DCNN (Pred.) as the second, red, line. Ad-

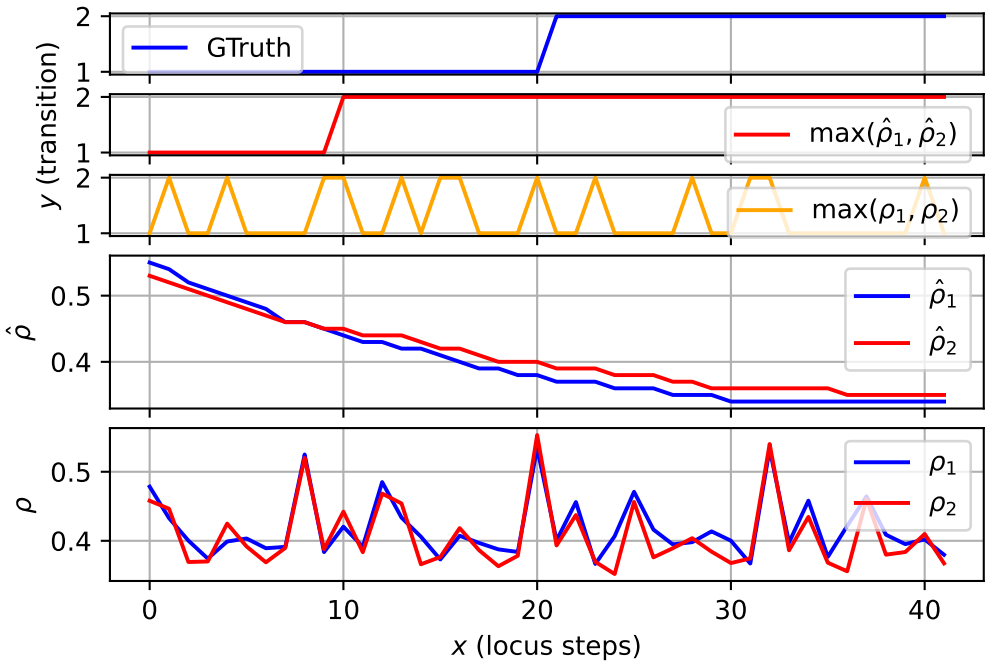

Figure 5.5: Device transition for a simple locus and different RT60, showing ideal device transition (GTruth), MSC predictor solution (max($\hat{\rho_1}, \hat{\rho_2}$)), the non-smoothed case (max(ρ_1, ρ_2)) and the relative MSC values used to derive transitions.

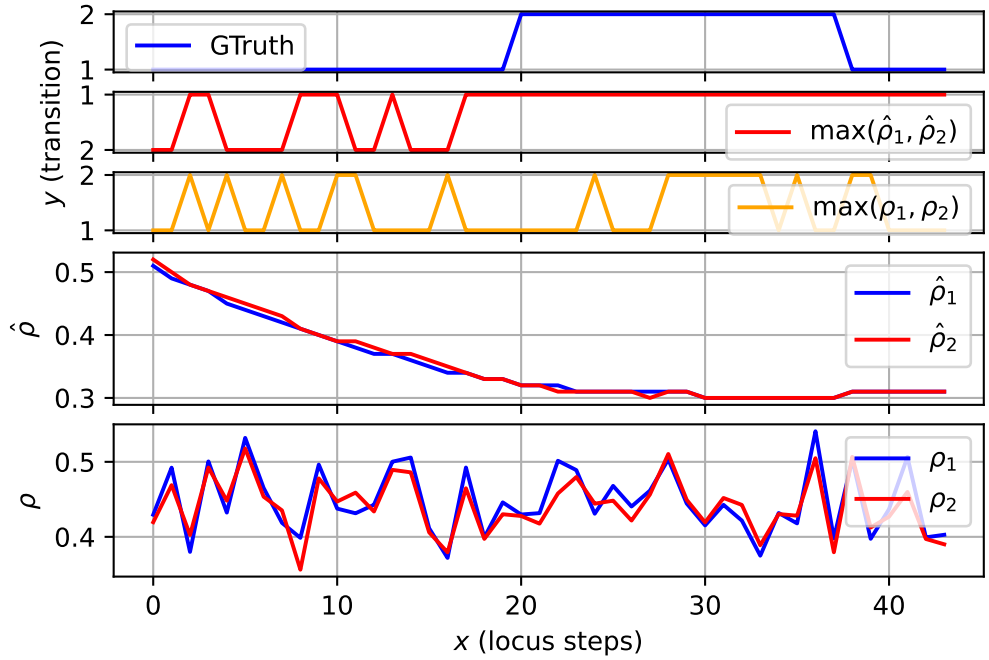

Figure 5.6: Device transition for a complex locus, showing ideal device transition (GTruth), MSC predictor solution (max($\hat{\rho_1}, \hat{\rho_2}$)), the non-smoothed case (max(ρ_1, ρ_2)) and the relative MSC values used to derive transitions.

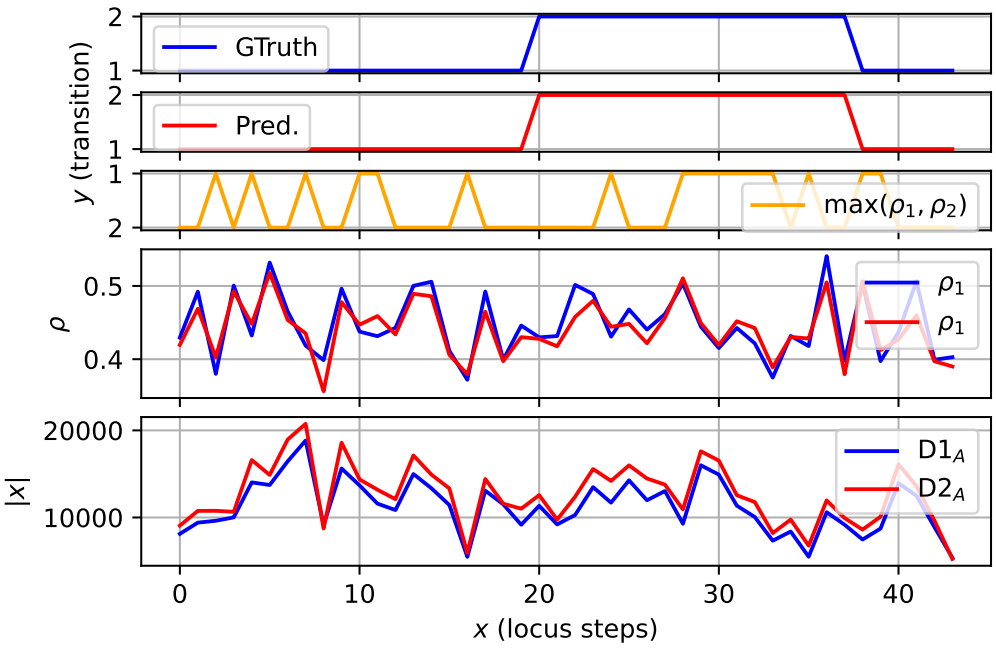

Figure 5.7: Device transition for a complex locus, showing ideal device transition (GTruth), proposed **1DCNN-CL** (Pred.), the non-predicted case (max(ρ_1, ρ_2)) and the relative MSC and A values used to predict the transitions.

ditionally, these graphs give more detail on the MSC used as a feature in the 1DCCN and as used in the Smoothed-MSC method. The MSC value is shown as the raw value from each device, ρ_1 and ρ_2 respectively, and, for the Smoothed-MSC comparative method, Figures 5.4, 5.5, 5.6, the doubly exponentially smoothed values are also shown as $\hat{\rho_1}$ and $\hat{\rho_2}$ respectively.

The first graph, Figure 5.4 depicts transitions based on the raw (non-smoothed) MSC values (max(ρ_1, ρ_2)) and the Smoothed-MSC transitions (max($\hat{\rho}_1, \hat{\rho}_2$)) when there is a simple locus. This shows that transitions are not clearly detected using raw MSC, whereas transitions using the Smoothed-MSC approach closely approximate the ideal scenario. However, when this technique is applied to a different room scenario in Figure 5.5 or more complex locus in Figure 5.6 we see that the Smoothed-MSC fails to follow the desired performance either transitioning at the wrong locus position in Figure 5.5 or with extraneous transitions in Figure 5.6.

The graph in Figure 5.7 illustrates the predicted transitions (Pred.) using the proposed 1DCNN with a custom loss function applied to the complex locus, closely approximating the ideal scenario during the transition. We only show one result for the

Table 5.4: Performance of the double exponential smoothing (S) compared with the Non-smoothed (NS) approach with different locus types and room RT60 values showing it can work with a simple locus (straight line) but fails with more complex loci.

		Linear locus		Random locus	
Metric	NS/S	RT60-0.8	RT60-1.5	RT60-0.8	RT60-1.5
WNE	NS	0.43	0.59	0.43	0.45
	S	0	0.08	0.64	0.66
ET	NS	20	22	23	26
	S	0	0	7	9

proposed technique in this diagram to save space as it worked also perfectly in the other cases. The graph displays raw MSC values (ρ_1, ρ_2) and signal magnitude A values ($D1_A$, $D2_A$) captured at each locus position, which are the features employed in the 1DCNN predictor for forecasting the transition.

Furthermore, we performed a quantitative analysis of these techniques across diverse locus movements, various rooms, and different audio signals. The assessment of quantitative performance relies on two primary metrics:

number of extraneous transitions (ET): The count of extra device transitions using a prediction compared to the number of device transitions using the ideal (ground truth) case. For example see Figure 5.6 which has seven extraneous transitions (six before the ground truth and one failed after).

weighted normalised error (WNE): uses the loss function $L(t)$ to compute an error that takes into account the fact that: while an error near the ideal switching point is benign, an error further away from this ideal switching point becomes perceptually highly significant (see Section 5.2).

Considering extraneous transitions as a metric is crucial since users generally would find unnecessary switches as very disruptive. Ideally, the number of extraneous transitions should be zero. The objective of WNE, is to calculate an error that increases when the selected device is substantially more distant than the ideal device and the ideal value should be zero.

The quantitative performance results shown in Table 5.4 highlight the poor performance of the comparative MSC predictor technique on different room conditions and on random locus movement. However, Table 5.5 shows that using our proposed 1DCNN

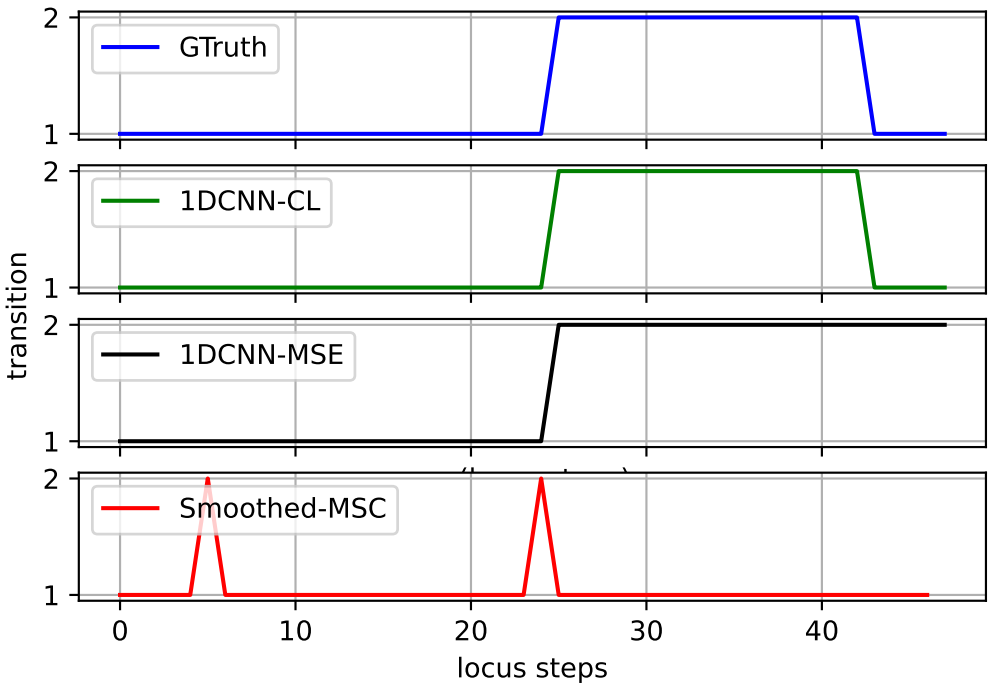

Figure 5.8: Samples from a single locus showing comparative results over a number of positions, using our proposed 1DCNN custom loss (**1DCNN-CL**) and the alternatives of either standard mean squared error as a loss function (1DCNN-MSE) or the exponentially smoothed MSC (Smoothed-MSC).

Table 5.5: Performance of proposed 1DCNN with custom loss (1DCNN-CL) with various features and compared against exponential smoothing (Smoothed-MSC) and a 1DCNN with a standard loss (1DCNN-MSE). Features: Absolute signal magnitude (A), MSC (C), multiple features (All). Complex loci used for all results.

Model	Features	WNE	ET
Smoothed-MSC	C	0.86	4783
1DCNN-MSE	A+C	0.22	1
1DCNN-CL	All	0.087	0
1DCNN-CL	C	0.05	0
1DCNN-CL	**A+C**	**0.03**	**0**

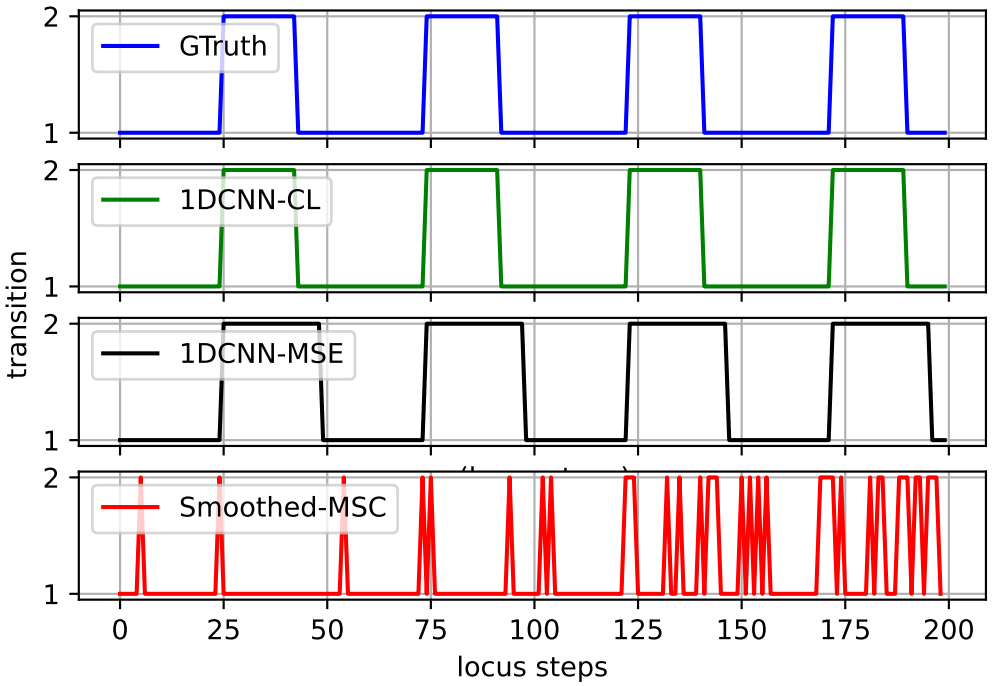

Figure 5.9: Samples from the results four loci (concatenated), showing comparative results for a number of positions, using our proposed 1DCNN custom loss (**1DCNN-CL**) and the alternatives of either standard mean squared error as a loss function (1DCNN-MSE) or the exponentially smoothed MSC (Smoothed-MSC).

.

solution significantly improves upon using the smoothed MSC approach, in particular exhibiting no extraneous transitions. Table 5.5 also explores different feature choices and loss function by training the model with different feature combinations: solely MSC (C), absolute signal magnitude (A), A with MSC (A+C), and all audio features described in Section 5.3.1.2 (All). We see that the custom loss (CL) function gives considerable improvement compared to a standard loss function such as MSE. We have chosen two results in Figures 5.8 and 5.9 to demonstrate the effect of the custom loss function, as we see the 1DCNN-CL more closely tracks the ground truth based upon the nearest device. Although the 1DCNN-CL does not transition exactly at the same point as the ground truth, as noted in Section 5.3.1, small differences are unlikely to be noticeable, however, with the standard MSE loss function (1DCNN-MSE) the device transition is much less accurate; indeed we see in Figures 5.8, 5.9 that one of the transitions was not detected at all for the standard loss function (1DCNN-MSE).

To demonstrate that the trained model can work across multiple rooms, as well as

Table 5.6: Performance of the proposed 1DCNN with custom loss (1DCNN-CL) using A+C features and with varied room type, Room 1 (RT1) and Room 2 (RT2) with various RT60 values.

Train RT1	Test RT2	WNE	ET
0.8s	0.8s	0.033	0
0.8s	1.5s	0.039	0
1.5s	0.8s	0.044	0

loci it was not trained for, the model was trained on Room 1 and then tested on Room 2 with different device positions and RT60 times. Table 5.6 shows this comparison with highly different room types RT indicating that the model performs well in different and with highly reverberating rooms (RT60=1.5s).

The overall performance analysis concludes that:

- The simple MSC predictor (Smoothed-MSC) offers a straightforward solution for detecting the appropriate smart speaker for handover, but it does not address the complexities of real-world loci scenarios.
- Using MSE as the loss function to train the 1DCNN model for smart speaker detection, improves the performance over the simple MSC predictor for loci of increased complexity, but is less accurate than the proposed solution and leads to missing/extraneous transitions
- The proposed **1DCNN-CL** model, trained with both the absolute signal magnitude and the MSC, addresses all complex real-world loci scenarios and produces optimal results – without including additional features such as the Mel magnitude spectrum or MFCCs.

5.5 Practical Considerations

5.5.1 Performance

Each convolution layer has $\mathcal{O}(nkfd)$ [81] complexity where n is the input size, k is the kernel size, f is the number of filters, and, d is the number of channels (number of features). Using the hypertuned model shown in Table 5.2 we thus find that for each

time sample the complete 1DCNN model would require of the order 0.7 M operations for each locus point. As there are 5 locus points per second this gives the number of operations as 3.5 M/s which is well within the capability of a small modern embedded processor such as would be found in a smart device.

5.5.2 Discussion

Another practical issue is expanding this work to more than two devices. This is more complex than the mechanism suggested in Section 4.4, in that chapter it is suggested that just the relative level of the MSC is used. Here, we do not have access to straightforward metrics within the model. Consequently, to operate with more that two devices it would be necessary to train with three, four or more devices and have a model for each. As the size of each model is less than one million parameters (from performance analysis immediately above) it should not be a difficult issue to store these models. However, it would require multiple training. In practice, it would be unusual to have more than four devices in one room[2] so that models for two, three and four devices should be sufficient.

5.6 Conclusions

This chapter presents a robust solution for seamless mid-call handovers between smart devices within a pervasive speech system, employing deep learning techniques. The presented solution builds upon the foundation established in the previous chapter, enhancing it to tackle real-world complexities; this addresses the research gap and questions on seamless handover between the devices in the same network i.e. horizontal handover as highlighted in Sections 1.3 and 2.8. The methodology involves the identification of the appropriate smart device through the processing of multivariate signalling features using 1DCNN and the utilisation of the SIP protocol for session transition. The research addresses challenges associated with employing 1DCNN as a classifier for device prediction and suggests a regression-based solution incorporating a custom loss function. The results indicate that using 1DCNN as a regressor with a custom loss function eliminates

[2] An Ofcom survey found only 6% of users had 5 or more smart speakers in their whole household [130].

unnecessary transitions and improves transition accuracy. This research also proposes a modified SIP signalling for an uninterrupted media transition. To validate the proposed 1DCNN with a custom loss, extensive testing was conducted in a simulated environment featuring diverse room conditions, varied talker movements, and a range of audio signals. This demonstrated that automatic switching between smart devices is possible without requiring exact calibration of the device locations and orientations.

6

Discussion and Conclusions

The solution presented in this thesis is, to the knowledge of the author, the first smart device handover mechanism for pervasive speech that does not require precise device calibration. Furthermore, it introduces novel solutions to optimise the session handover latency for seamless media transmission at the application layer for both network and device session transitions. This represents a significant advancement, but it is evident that further efforts would be necessary to develop a fully functional system based on the proposal presented in this thesis. We consider some of these aspects here, first considering the contributions, followed by an exploration of practical issues, future considerations, and our final conclusions.

6.1 Contributions of the Thesis

As the Internet and heterogeneous communication networks continue to advance, coupled with the increasing popularity of various smart devices like smartphones, smart speakers, and other communication platforms, there arises a necessity to ensure a unified speech communication experience provided by these technologies and devices. As pointed out in Chapter 1, Despite these evolving trends, minimal effort has been made to unify or simplify the user experience across multiple devices and environments, leading to users still dealing with 'non-pervasive' communication experience.

However, as reviewed in Chapter 2, there are promising indications within the industry towards addressing this issue and moving towards providing solutions for a 'pervasive' communication experience. Companies such as Amazon and Apple have introduced features like 'group chat', HomePod *etc.* solutions to their respective smart speaker offerings. While these solutions are steps closer to achieving a 'pervasive' experience, they still fall short of fulfilling the requirements of pervasive speech, particularly in scenarios where two-way communication is not feasible with the aforementioned industry solutions.

In academia, the author did not come across any significant research specifically addressing the pervasive speech experience. These observations and existing gaps in the literature motivated the author to undertake this research and explore solutions for seamless network and device handovers, aimed at achieving a pervasive speech experience.

Chapter 3 explores the existing gaps and offers solutions for achieving seamless vertical handover, specifically focusing on ensuring a smooth transition of media in network handover scenarios such as a user being connected to an LTE network while driving a car and subsequently connecting to a Wi-Fi network upon entering a house, or vice versa. Research findings outlined in Section 3.3.1 highlight the potential for latency during network session handovers, which can lead to disruptions in media. However, this issue can be addressed by predicting network transitions and implementing proactive

session handovers. The proposed solution of using a supervised machine learning technique for predicting the network transitions at the application layer using SIP terminal mobility feature, as described in Section 3.2.1 and the comparative results detailed in Section 3.3.4 validate the effectiveness of the proposed solution in achieving seamless vertical handover.

Chapter 4 and Chapter 5 present seamless solutions for device handover, aiming to enhance the pervasive communication experience. These solutions cater to scenarios where a user transitions from using a smartphone to a smart device upon entering a house and subsequently switches between devices within the same network domain. These device transitions primarily involve two steps: device detection, followed by session handover to the detected device.

Regarding the seamless session handover during device transition within the same network, this study examines two SIP session handover methods: leveraging existing SIP features like personal mobility via sequential or parallel call forking, and utilising personal mobility with modified SIP functionality. From the signal flow outlined in Section A.2.1, it is evident that the first approach could lead to interruptions in media flow, and the corresponding node (CN) would need to be aware of device transitions, which may not be essential and overburdens the signalling flow. However, this approach doesn't necessitate any modifications to SIP signalling. Therefore, with this signalling method, only managing device detection is necessary as a comprehensive solution for device transitions. The modified SIP solution for session handover as detailed in Section A.2.2, offers benefits such as no disruptions in media flow, and device transitions are managed at the home gateway B2BUA without the involvement of the corresponding node (CN). Due to these advantages, this study advocates for adopting this modified SIP solution for session handover between the devices.

In terms of device detection, this research explores the techniques to detect the device based on audio signal features, eliminating the need for calibration. This approach marks a pioneering effort in leveraging audio features for this specific use case, drawing inspiration from previous research on location-based technologies. The decision not to

use calibration is motivated by the challenges associated with its use, as device detection accuracy may be compromised when devices are moved or their orientation changes, circumstances commonly encountered in indoor environments.

This thesis highlights the importance of the MSC as a prominent audio signal feature for device detection, a notion supported by existing literature on sound source localisation techniques (Please refer to Section 2.6). A device exhibiting high MSC is a suitable device for handover. However, it is observed that the captured MSC values are influenced by room reflections, necessitating the use of time-series predictors to predict the raw MSC readings. The comparison results provided in Section 4.3.2 emphasize the efficacy of MSC prediction utilising double exponential smoothing, demonstrating superior performance when compared to other time-series predictors like AR, MR, ARMA, and ARIMA.

The MSC prediction using double exponential smoothing serves as a fundamental solution for device detection without the use of calibration. However, the results discussed in Section 5.4.4 show that only relying on the MSC as an isolated feature is insufficient. Its efficacy is limited, particularly when confronted with real-world scenarios that involve user locus movements and the influence of room reflections (RT60), which impact the raw MSC values. To deal with this problem and cover a wider range of user locus movements and different room conditions, adoption of advanced deep learning techniques becomes essential as a pivotal solution. Chapter 5 details the device detection based on deep learning techniques. The author adopted 1DCNN deep learning technique, as suggested by the literature survey (Please refer to Section 2.6). An extensive a hyperparameter tuning process was conducted using the custom loss function to identify the number of convolutional layers, filters, and units to achieve optimal performance. The experiments involved training the model using several multivariate audio signal features, including MSC, signal magnitude (A), and MFCC values, along with a custom loss function, to account for various user movements and utilize multiple audio signals representing distinct sound sources. In-depth analysis revealed that training the model with MSC and signal magnitude yielded optimal outcomes. Consequently, the research concludes that

this approach stands as a robust solution for device detection.

6.2 Further Considerations

6.2.1 Device detection

This research offers two solutions for device detection in horizontal handover. The first solution is the MSC predictor, which employs a double exponential smoothing technique to address basic linear user movements. The second solution is more robust, utilising machine learning techniques, and effectively handling real-world complexities, such as variations in rooms and reverberation characteristics. For the robust solution, the machine learning would need to be optimised across a much wider set of examples. For example, while the simulation approach used in this thesis shows good proof of principle, further training in real environments would help train the system across a wider set of conditions. This would require a method of tracking a talker while taking part in the training sessions. Advanced techniques for user tracking using video have been proven in other fields for machine learning purposes. For example, the use of eye-tracking to record user interactions with Video-on-Demand (VoD) applications in order to understand user experience [131]. Advanced video-based user tracking techniques could be employed to provide the required ground-truth inputs from training sessions with minimal operational effort.

Additionally, the training in this thesis was specific to the microphone placement in the smart devices, we suspect that the system would need retraining for different microphone placements i.e., training is likely to be required for each type of device.

In this thesis, only one talker was used in the simulations. While this is a good scenario for domestic use cases, as often there is one dominant talker in a location, this would be less valid in situations where there are multiple talkers in a single auditory environment such as business meetings with maybe only one or two remote members. It may be in such cases the switching of the talker input proposed in this thesis would be useful to use the smart device closest to the talker, but switching the played back speech

into the room may not benefit from switching as it would make the remote talker's apparent location in the room jump from one device to another. However this issue can be mitigated with the use of echo cancellation on smart speakers [132]. Further work on this user experience for different meeting types is thus required. Additionally, background noise sources (such as a television or washing machine in a domestic situation) will impact the performance and an engineering solution would be required to track the speech of the talker rather than the background noise. This could be achieved using solutions such as blind-source separation [133], however, there would need to be some expert system to decide which source is the best to track for the application.

In the early stages of exploring deep learning techniques for device detection in handling the real-world complexities, the author initially explored Long Short-Term Memory (LSTM) for the device prediction but initial experiments did not achieve good results. This led to the adoption of a 1DCNN solution, which yielded promising results. Although this was not discussed in previous chapters of the thesis, the application of LSTM and other machine learning techniques remains a potential area for future research. In particular, LSTM is a promising alternative technique and is widely used by itself or in combination with a CNN. Although a form of LSTM has been used (with a CNN) for sound localisation [134], it should be noted that this thesis considers tracking a moving system where the most recent location is the most important factor; consequently, too much dependence on long-term data could lead to erroneous results, this may have been the reason that early experiments with LSTM, on their own, were not promising. Potentially, a combination of both LSTM and a 1DCNN would be an interesting study for further work.

Finally, we commented in Section 5.2 that we assumed the device nearest the user would be the ideal device. The use of coherence in the approach is an interesting feature. We conjecture that in a room that consists of a portion with a dead acoustic environment (such as that set up for living space with carpets) and a portion with a live acoustic environment (such as a kitchen with hard surfaces), it would be beneficial to switch to a device in the dead environment earlier than that in the live environment. Thus, it might

be that the use of a coherence-based ground truth rather than a distance-based ground truth will result in improved perceptual performance. This would require subjective testing which was out of the scope of work for this thesis.

6.2.2 Session handling

This research suggests modifications to SIP signalling to facilitate seamless session handovers. However, it is important to note that only theoretical proposals are outlined and practical implementation of the proposed solution has not been carried out as a scope of this thesis. These proposals are based on the author's experience with SIP, gained from working in SIP call server systems at BT. To carry out the verification of the signalling solution in lab environment, an engineering team is essential to navigate the complexities involved and all of the required practicalities are out of the scope of a PhD thesis such as this.

Regarding the proposed signalling solution for network handovers, as discussed in Section 3.2, wasn't feasible to verify in a lab environment. This is because the suggested signalling handover, according to ML predictions, needs to occur at the network level, making it challenging to simulate the network handover scenario in a lab environment. However, verifying this solution could be achievable with a few weeks of effort and support from an engineering team in an environment where network handovers can be adjusted, such as in the BT lab environment. For these reasons, the proposed solution was validated using real-world mobility data, as described in Section 3.2.2.1.

Regarding the solution for signalling in the case of horizontal device handover, as elaborated in Section 4.2.1 and briefly outlined in Section 5.3.2, since the proposed signalling changes can be carried at the home/office gateway level, practical validation can be conducted using Asterisk, an open-source SIP call server, with several weeks of effort from an engineering team. However, as the author currently lacks engineering support, this practical validation is beyond the scope of this thesis and is proposed as a future improvement.

6.3 Conclusions

In summary, this thesis has delved into the complexities of achieving seamless handover for pervasive speech communication, illuminating crucial elements such as device detection based on audio features and the seamless execution of vertical and horizontal session handovers using the application layer SIP protocol. Through rigorous research and analysis, we have proposed solutions involving time-series prediction algorithms and deep learning techniques to enhance the detection of suitable devices, thereby improving overall performance. Additionally, our work addresses challenges related to preventing media interruptions during SIP session handovers. Despite the encountered challenges, the research journey has been highly rewarding, with the acquired knowledge in audio signal processing and deep learning laying the groundwork for future explorations and advancements in the pervasive speech system.

The culmination of this research holds significant implications for the smart speaker market, emphasizing the need for continued investigation and innovation in the realm of converged communications experience. Upon reflection of accomplishments and lessons learned, it becomes apparent that raw speech signal features exhibit notable variations due to environmental factors, particularly room effects. Consequently, this study underscores the necessity of the predictors to significantly reduce extraneous transitions that would otherwise result in a poor user experience. Three solutions are introduced: a basic smoothing predictor (MSC predictor), prediction through One-Dimensional Convolutional Neural Networks (1DCNN) with standard mean squared error as the error function, and IDCNN utilising a custom loss function as the error function. The results indicate that using 1DCNN with a custom loss function eliminates unnecessary transitions and improves transition accuracy.

The collaborative efforts and the unwavering support from my mentor and also the delegates from BT have been instrumental in the success of this thesis. Moving forward, I hope that the insights gained here will inspire further inquiry and contribute to the ongoing evolution of converged communications experience.

A

Appendix A

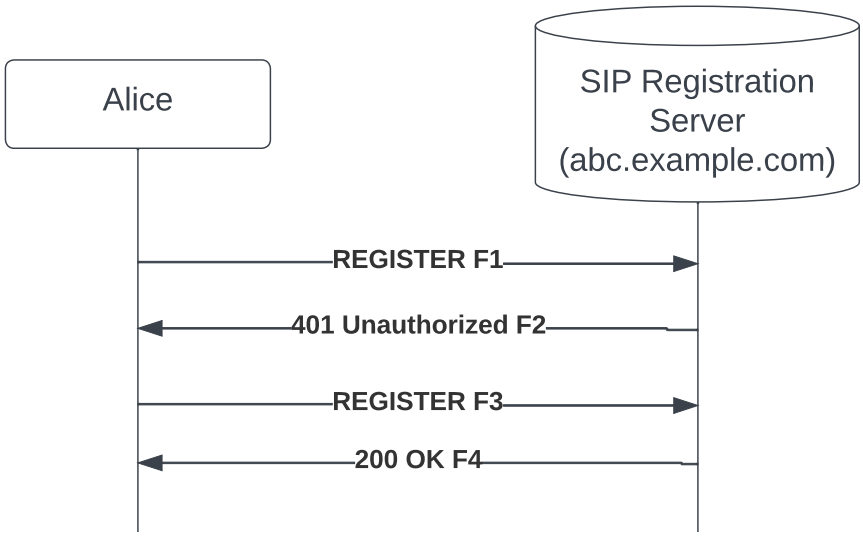

Figure A.1: First time Registration

A.1 Message sequence to register and session establishment on devices

A.1.0.1 Message sequence for new registration

```
F1 REGISTER Alice -> SIP Server

 REGISTER sip:ss2.abc.example.com SIP/2.0
 Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashds7
 Max-Forwards: 70
 From: Alice <sip:alice@abc.example.com>;tag=a73kszlfl
 To: Alice <sip:alice@abc.example.com>
 Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
 CSeq: 1 REGISTER
 Contact: <sip:alice@mobile.abc.example.com>
 Content-Length: 0

F2 401 Unauthorized SIP Server -> Alice

 SIP/2.0 401 Unauthorized
 Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashds7;received=192.0.2.201
 From: Alice <sip:alice@abc.example.com>;tag=a73kszlfl
```

```
To: Alice <sip:alice@abc.example.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="atlanta.example.com", qop="auth",
nonce="ea9c8e88df84f1cec4341ae6cbe5a359",
opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0

F3 REGISTER Alice -> SIP Server

REGISTER sip:ss2.abc.example.com SIP/2.0
Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashd92
Max-Forwards: 70
From: Alice <sip:alice@abc.example.com>;tag=ja743ks76zlflH
To: Alice <sip:alice@abc.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
CSeq: 2 REGISTER
Contact: <sip:alice@mobile.abc.example.com>
Authorization: Digest username="alice", realm="atlanta.example.com"
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:ss2.abc.example.com",
response="dfe56131d1958046689d83306477ecc"
Content-Length: 0

F4 200 OK SIP Server -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashd92;received=192.0.2.201
From: Alice <sip:alice@abc.example.com>;tag=ja743ks76zlflH
To: Alice <sip:alice@abc.example.com>;tag=37GkEhwl6
Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
CSeq: 2 REGISTER
Contact: <sip:alice@mobile.abc.example.com>;expires=3600
Content-Length: 0
```

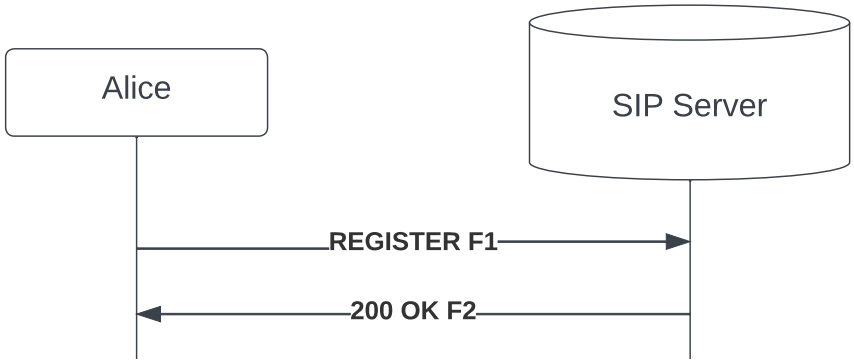

Figure A.2: Update Registration with new contacts

A.1.0.2 Message sequence to update registration

```
F1 REGISTER Alice -> SIP Server

 REGISTER sip:ss2.abc.example.com SIP/2.0
 Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashds7
 Max-Forwards: 70
 From: Alice <sip:alice@abc.example.com>;tag=a73kszlfl
 To: Alice <sip:alice@abc.example.com>
 Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
 CSeq: 1 REGISTER
 Contact: <sip:alice@speaker.abc.example.com>
 Authorization: Digest username="alice", realm="atlanta.example.com",
 qop="auth", nonce="1cec4341ae6cbe5a359ea9c8e88df84f", opaque="",
 uri="sip:ss2.abc.example.com",
 response="71ba27c64bd01de719686aa4590d5824"
 Content-Length: 0

 F2 200 OK SIP Server -> Alice

 SIP/2.0 200 OK
 Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
 From: Alice <sip:alice@abc.example.com>;tag=a73kszlfl
 To: Alice <sip:alice@abc.example.com>;tag=34095828jh
 Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
 CSeq: 1 REGISTER
```

```
Contact: <sip:alice@mobile.abc.example.com>;expires=3600
Contact: <sip:alice@speaker.abc.example.com>;expires=4294967295
Content-Length: 0

F1 REGISTER Alice -> SIP Server

REGISTER sip:ss2.abc.example.com SIP/2.0
Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Alice <sip:alice@abc.example.com>;tag=a73kszlfl
To: Alice <sip:alice@abc.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
CSeq: 1 REGISTER
Contact: <sip:alice@pc.abc.example.com>
Authorization: Digest username="alice", realm="atlanta.example.com",
qop="auth", nonce="1cec4341ae6cbe5a359ea9c8e88df84f", opaque="",
uri="sip:ss2.abc.example.com",
response="71ba27c64bd01de719686aa4590d5824"
Content-Length: 0

F2 200 OK SIP Server -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/TLS mobile.abc.example.com:5061;branch=z9hG4bKnashds7
;received=192.0.2.201
From: Alice <sip:alice@abc.example.com>;tag=a73kszlfl
To: Alice <sip:alice@abc.example.com>;tag=34095828jh
Call-ID: 1j9FpLxk3uxtm8tn@abc.example.com
CSeq: 1 REGISTER
Contact: <sip:alice@mobile.abc.example.com>;expires=3600
Contact: <sip:alice@speaker.abc.example.com>;expires=4294967295
Contact: <sip:alice@pc.abc.example.com>;expires=4294967295
Content-Length: 0
```

A.1.0.3 Message sequence to setup session on multiple devices

```
1 INVITE CN -> B2BUA
```

```
 INVITE sip:alice@abc.example.com SIP/2.0
 Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9
 Max-Forwards: 70
 From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
 To: Alice <sip:alice@abc.example.com>
 Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
 CSeq: 1 INVITE
 Contact: <sip:cn@mobile.xyz.example.com>
 Content-Type: application/sdp
 Content-Length: 151

2 100 Trying B2BUA -> CN

 SIP/2.0 100 Trying
 Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9 ;received=192.0.2.101
 From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
 To: Alice <sip:alice@abc.example.com>
 Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
 CSeq: 1 INVITE
 Content-Length: 0

 3a INVITE B2BUA -> Alice (M1)

 INVITE sip:alice@M1.abc.example.com SIP/2.0
 Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
 Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
 Max-Forwards: 69
 From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
 To: Alice <sip:alice@abc.example.com>
 Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
 CSeq: 1 INVITE
 Contact: <sip:cn@mobile.xyz.example.com>
 Content-Type: application/sdp
 Content-Length: 151
```

```
3b INVITE B2BUA -> Alice (S1)

INVITE sip:alice@S1.abc.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards: 69
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:cn@mobile.xyz.example.com>
Content-Type: application/sdp
Content-Length: 151

3c INVITE B2BUA -> Alice (PC1)

INVITE sip:alice@PC1.abc.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards: 69
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:cn@mobile.xyz.example.com>
Content-Type: application/sdp
Content-Length: 151


4a 180 Ringing Alice (M1) -> B2BUA

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.222
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9 ;received=192.0.2.101
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
```

```
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@M1.abc.example.com>
Content-Length: 0

4b 180 Ringing Alice (S1) -> B2BUA

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.222
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9 ;received=192.0.2.101
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@S1.abc.example.com>
Content-Length: 0

4c 180 Ringing Alice (PC1) -> B2BUA

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.222
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9 ;received=192.0.2.101
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@PC1.abc.example.com>
Content-Length: 0

5 180 Ringing B2BUA -> CN

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9 ;received=192.0.2.101
```

```
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@M1.abc.example.com>
Content-Length: 0

6a 200 OK Alice (M1) -> B2BUA

SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1 ;received=192.0.2.222
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9  ;received=192.0.2.101
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@M1.abc.example.com>
Content-Type: application/sdp
Content-Length: 147

6b 200 OK Alice (S1) -> B2BUA

SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1 ;received=192.0.2.222
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9  ;received=192.0.2.101
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@S1.abc.example.com>
Content-Type: application/sdp
Content-Length: 147

6c 200 OK Alice (PC1) -> B2BUA

SIP/2.0 200 OK
```

```
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1 ;received=192.0.2.222
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9  ;received=192.0.2.101
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@PC1.abc.example.com>
Content-Type: application/sdp
Content-Length: 147

7 200 OK B2BUA -> CN

SIP/2.0 200 OK
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9 ;received=192.0.2.101
From: CN <sip:cn@abc.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 INVITE
Contact: <sip:alice@M1.abc.example.com>
Content-Type: application/sdp
Content-Length: 147

8 ACK CN -> B2BUA

ACK sip:alice@abc.example.com SIP/2.0
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
Contact: <sip:cn@mobile.xyz.example.com>
CSeq: 1 ACK
Content-Length: 0

9a ACK B2BUA -> Alice (M1)
```

```
ACK ip:alice@M1.abc.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards: 69
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 ACK
Contact: <sip:cn@mobile.xyz.example.com>
Content-Length: 0
```

```
9b ACK B2BUA -> Alice (S1)
```

```
ACK ip:alice@S1.abc.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards: 69
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 ACK
Contact: <sip:cn@mobile.xyz.example.com>
Content-Length: 0
```

```
9c ACK B2BUA -> Alice (PC1)
```

```
ACK ip:alice@PC1.abc.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.abc.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP mobile.xyz.example.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards: 69
From: CN <sip:cn@xyz.example.com>;tag=9fxced76sl
To: Alice <sip:alice@abc.example.com>
Call-ID: 2xTb9vxSit55XU7p8@xyz.example.com
CSeq: 1 ACK
Contact: <sip:cn@mobile.xyz.example.com>
Content-Length: 0
```

```
Alice - CN
Proxy 2 - B2BUA
Bob - Alice

biloxi - abc
atlanta - xyz
client - mobile
```

A.2 SIP signalling for pervasive device handover

A.2.1 Session handover with existing SIP

The detailed SIP message flow during device session handover is shown in Figure. A.3. For the full registration messages, please see Appendix A.1. Please note that the figure depict only the signalling flow, omitting the media-related message flow. In this process, using an out-of-band mechanism, B2BUA sends a Re-Invite/REFER SIP message to the second device for session handover. No modifications are needed in the existing SIP stack for this purpose. However, it is worth noting that prior research has indicated that successful session establishment or re-establishment may take at least 280 milliseconds, resulting in a temporary interruption to the media during session handovers [23]. Also, the corresponding node (CN) is involved in this session transition. Our research is driven by these limitations to propose an alternative approach for session handover.

A.2.2 Session handover with modified SIP

The detailed SIP message flow during session handover with this proposed method is shown in Figure. A.4. In this alternative approach, again, an out-of-band mechanism will initiate B2BUA to have proactive parallel sessions on all the devices that are within the

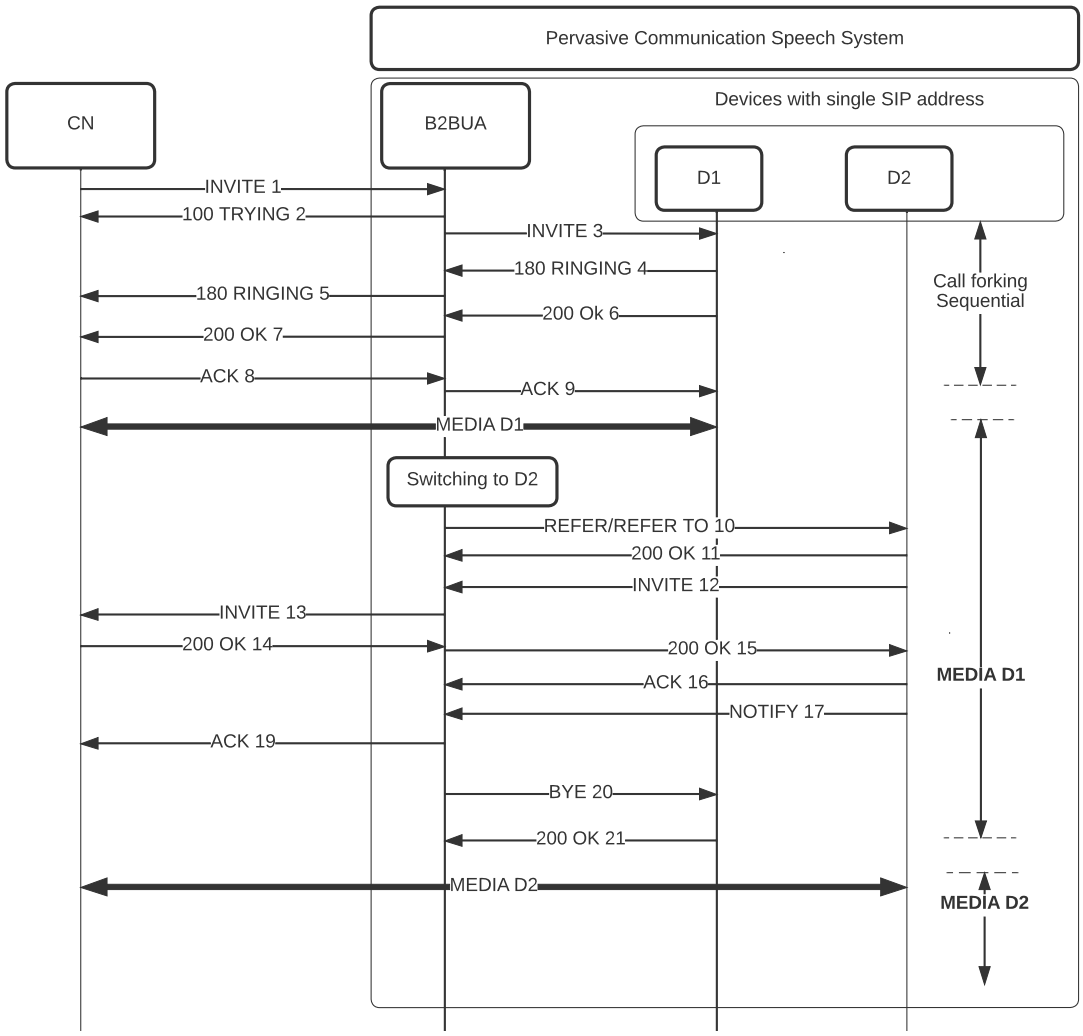

Figure A.3: SIP Session handover using sequential call forking, and CN is aware of device change. Also note the short breaks in the media transmission.

pervasive application domain. To achieve this session establishment, the system employs *parallel call forking*. In the case of *sequential call forking*, when a session is established on one device, the B2BUA refrains from initiating any SIP messages for session establishment on other devices. However, in the context of *parallel call forking*, standard SIP behavior dictates that the B2BUA cancels the Invites sent to other devices once session on one device is established. For this paper's requirements, which entail parallel sessions on all devices to ensure media continuity, a modification to the B2BUA's behavior is necessary. Specifically, after a session is established on one device, the B2BUA should not cancel the Invites that were sent to other devices. These behavioral adjustments

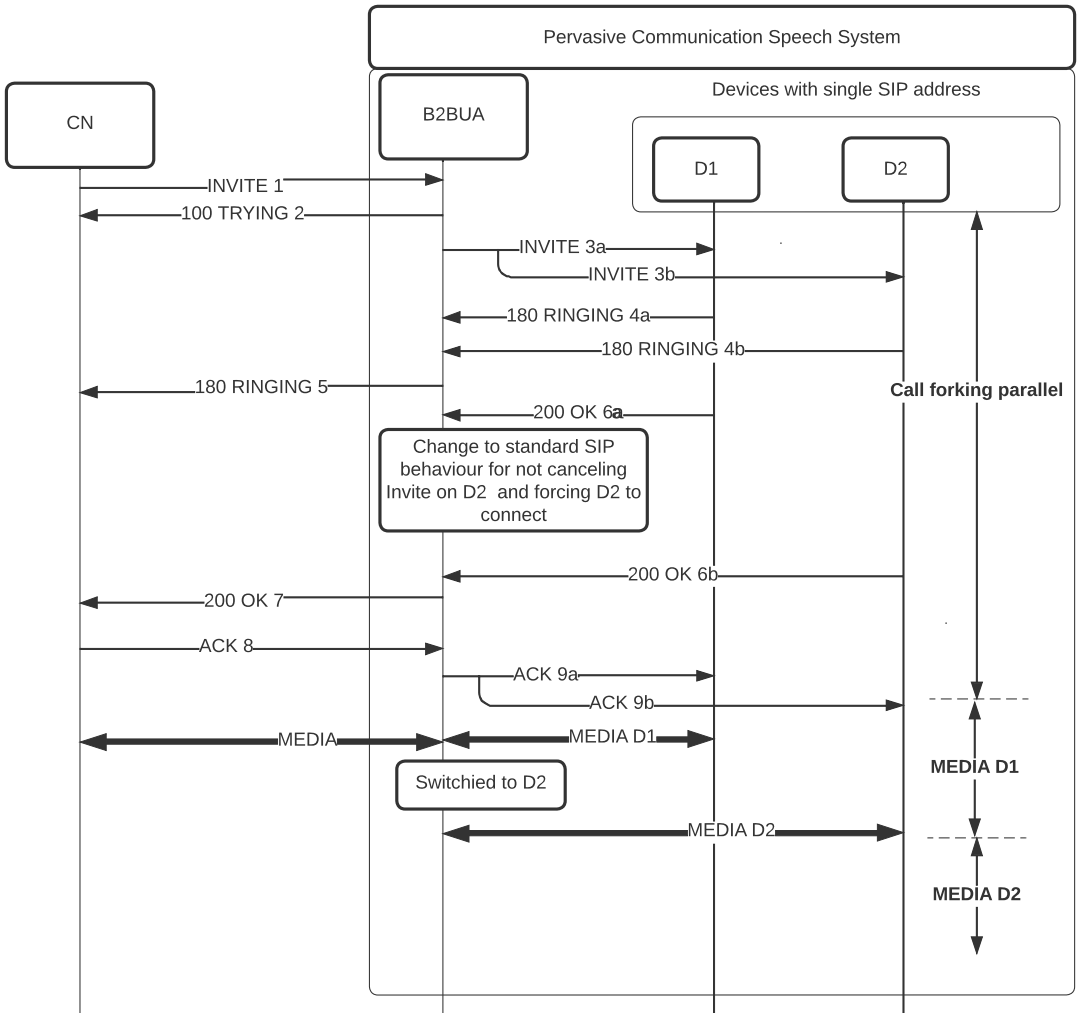

Figure A.4: SIP Session handover using modified approach, and CN is not aware of device change. Also note the seamless media transmission during the device transition.

can be implemented using SIP header extensions or custom headers. Importantly, these changes are limited to the handover domain, such as smart devices within a home environment; therefore, there is no need for modifications to remote SIP systems. The health of the devices participating in the session can be monitored using standard SIP monitoring message support such as OPTIONS.

References

[1] Advancing ICT Industry Transformation. "Advancements in Industry Technologies." (2023), [Online]. Available: https://www.atis.org/wp-content/uploads/2023/02/2023-Overview-2-4.pdf.

[2] C. K. Wu, C.-T. Cheng, Y. Uwate, G. Chen, S. Mumtaz, and K. F. Tsang, "State-of-the-Art and Research Opportunities for Next-Generation Consumer Electronics," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 937–948, 2023. DOI: 10.1109/TCE.2022.3232478.

[3] "What makes a smart home smart?" (2023), [Online]. Available: https://www.globalservices.bt.com/en/insights/whitepapers/what-makes-a-smart-home-smart.

[4] Ericsson. "Mobile data traffic outlook," Industry Insights. (2023), [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast.

[5] Mordor Intelligence. "Smartphone Market Size." (Accessed 21st August 2024), [Online]. Available: https://www.mordorintelligence.com/industry-reports/smartphones-market.

[6] Statista. "Smartphones - Worldwide." (2024), [Online]. Available: https://www.statista.com/outlook/cmo/consumer-electronics/telephony/smartphones/worldwide.

[7] 3GPP, "Architecture enhancements for non-3GPP accesses," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.402, 2019..

[8] M. Humphries. "Amazon Adds Group Chat to Echo Devices," Industry Insights. (2020), [Online]. Available: https://www.pcmag.com/news/amazon-adds-group-chat-to-echo-devices.

[9] Amazon. "Amazon Adds two-way conversation to Alexa enabled devices." (2022), [Online]. Available: https://www.amazon.com/gp/help/customer/display.html?nodeId=GS3WRTSRKD2U6MCK.

[10] Apple. "Apple HomePod," Industry Insights. (2023), [Online]. Available: https://support.apple.com/en-gb/guide/homepod/apdeaa15a6c3/homepod.

[11] E. H. Schwartz. "Deutsche Telekom is Launching an Entry-Level Smart Speaker With Both Magenta and Alexa." (Jun. 2020), [Online]. Available: https://voicebot.ai/2020/06/10/deutsche-telekom-is-launching-an-entry-level-smart-speaker-with-both-magenta-and-alexa.

[12] Y. Xiong, X. Tang, S. Li, *et al.*, "mmAcoustic: Full-Field Sound Source Localization, Identification, and Area-Selectable Sound Recovery via Millimeter-Wave Vibration Monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–12, 2024. DOI: 10.1109/TIM.2023.3342841.

[13] W. Maass, J. Parsons, S. Purao, V. Storey, and C. Woo, "Data-Driven Meets Theory-Driven Research in the Era of Big Data: Opportunities and Challenges for Information Systems Research," *Journal of the Association for Information Systems*, Jun. 2018. DOI: 10.17705/1jais.00526.

[14] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2014. DOI: 10.1109/TMC.2013.27.

[15] S. Uppoor and M. Fiore, "Characterizing Pervasive Vehicular Access to the Cellular RAN Infrastructure: An Urban Case Study," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2603–2614, 2015. DOI: 10.1109/TVT.2014.2343651.

[16] R. Scheibler, E. Bezzam, and I. Dokmanic, "Pyroomacoustics: A Python Package for Audio Room Simulation and Array Processing Algorithms," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Apr. 2018. DOI: 10.1109/icassp.2018.8461310. [Online]. Available: https://doi.org/10.1109%2Ficassp.2018.8461310.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964.

[18] R. M. Corey, M. D. Skarha, and A. C. Singer, *Massive Distributed Microphone Array Dataset*, 2019. DOI: 10.13012/B2IDB-6216881_V1. [Online]. Available: https://doi.org/10.13012/B2IDB-6216881_V1.

[19] M. D. Corey Ryan M.and Skarha and A. C. Singer, "Cooperative Audio Source Separation and Enhancement Using Distributed Microphone Arrays and Wearable Devices," in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2019, pp. 296–300. DOI: 10.1109/CAMSAP45676.2019.9022475.

[20] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python*, 2001–. [Online]. Available: http://www.scipy.org/.

[21] B. McFee, C. Raffel, D. Liang, *et al.*, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 2011.

[23] F. Chollet *et al.* "Keras." (2015), [Online]. Available: https://github.com/fchollet/keras.

[24] M. Philpott. "The Future Telco Connected Home." (Jan. 2021), [Online]. Available: https://www.broadband-forum.org/.

[25] "The foundation for connected things." (Accessed 21st August 2024), [Online]. Available: https://csa-iot.org/all-solutions/matter/.

[26] Y. Körber, M. Feld, and T. Schwartz, "Pervasive audio playback in cyber-physical environments," in *2017 Intelligent Systems Conference (IntelliSys)*, 2017, pp. 531–541. DOI: 10.1109/IntelliSys.2017.8324346.

[27] Y. Körber, "Device selection for speech output in a multi-user scenario," in *Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '17, Stuttgart, Germany: Association for Computing Machinery, 2017, pp. 363–370, ISBN: 9781450353786. DOI: 10.1145/3152832.3156624. [Online]. Available: https://doi.org/10.1145/3152832.3156624.

[28] A. B. Johnston, *SIP: understanding the session initiation protocol*. Artech House, 2015.

[29] E. Schooler, J. Rosenberg, H. Schulzrinne, *et al.*, *SIP: Session Initiation Protocol*, IETF RFC 3261, Jul. 2002. DOI: 10.17487/RFC3261. [Online]. Available: https://rfc-editor.org/rfc/rfc3261.txt.

[30] A. B. Johnston, S. Donovan, R. Sparks, C. Cunningham, and K. Summers, *Session initiation protocol (SIP) basic call flow examples*, IETF RFC 3665, 2003. DOI: 10.17487/RFC3665. [Online]. Available: https://doi.org/10.17487/RFC3665.

[31] J. D. Rosenberg, *A Session Initiation Protocol (SIP) Event Package for Registrations*, IETF RFC 3680, 2004. DOI: 10.17487/RFC3680. [Online]. Available: https://doi.org/10.17487/RFC3680.

[32] S. Olson, G. Camarillo, and A. Roach, *Support for IPv6 in Session Description Protocol (SDP)*, IETF RFC 3266, 2002. DOI: 10.17487/RFC3266. [Online]. Available: https://doi.org/10.17487/RFC3266.

[33] M. A. Qadeer, A. H. Khan, J. A. Ansari, and S. Waheed, "IMS network architecture," *Proc. - 2009 Int. Conf. Futur. Comput. Commun. ICFCC 2009*, pp. 329–333, 2009. DOI: 10.1109/ICFCC.2009.106.

[34] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP," *IEEE Globecom 2000 Work. - 2000 IEEE Serv. Portability Virtual Cust. Environ.*, pp. 29–36, 2000. DOI: 10.1109/SPVCE.2000.934158.

[35] R. Mahmood and M. A. Azad, "SIP messages delay analysis in heterogeneous network," in *2010 International Conference on Wireless Communication and Sensor Computing (ICWCSC)*, 2010, pp. 1–5. DOI: 10.1109/ICWCSC.2010.5415913.

[36] E. S. Boysen and J. Flathagen, "Using SIP for seamless handover in heterogeneous networks," in *2011 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2011, pp. 1–8.

[37] T. Hemmati and M. H. Y. Moghadam, "SIP-based vertical handover scheme with bicasting," in *16th International Conference on Advanced Communication Technology*, 2014, pp. 19–22. DOI: 10.1109/ICACT.2014.6778914.

[38] J. C. Vijayshree and T. G. Palanivelu, "Vertical handover triggering between WLAN and WIMAX using SIP," *Proc. 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2014*, pp. 769–774, 2015. DOI: 10.1109/ICACCCT.2014.7019195.

[39] Z. Tsiatsikas, A. Fakis, D. Papamartzivanos, D. Geneiatakis, G. Kambourakis, and C. Kolias, "Battling against DDoS in SIP: Is Machine Learning-based detection an effective weapon?" In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, vol. 04, 2015, pp. 301–308.

[40] K. Umoto, S. Ata, Y. Chimura, N. Nakamura, and T. Yao, "An Automatic Error Identification Method in Call Control Protocol Using Levenshtein Distance," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2020, pp. 266–273. DOI: 10.1109/ICIN48450.2020.9059524.

[41] D. Pereira, R. Oliveira, and H. S. Kim, "A Machine Learning Approach for Prediction of Signaling SIP Dialogs," *IEEE Access*, vol. 9, pp. 44 094–44 106, 2021. DOI: 10.1109/ACCESS.2021.3065660.

[42] S. Toufga, S. Abdellatif, P. Owezarski, T. Villemur, and D. Relizani, "Effective Prediction of V2I Link Lifetime and Vehicle's Next Cell for Software Defined Vehicular Networks: A Machine Learning Approach," in *2019 IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–8. DOI: 10.1109/VNC48660.2019.9062803.

[43] N. Nayakwadi and R. Fatima, "Machine learning based handover execution algorithm for heterogeneous wireless networks," in *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2020, pp. 54–58. DOI: 10.1109/ICRCICN50933.2020.9296169.

[44] S. Memon and M. Maheswaran, "Using machine learning for handover optimization in vehicular fog computing," *Proc. ACM Symp. Appl. Comput.*, vol. Part F1477, 2019. DOI: 10.1145/3297280.3297300.

[45] K. Kousias, Ö. Alay, A. Argyriou, A. Lutu, and M. Riegler, "Estimating Downlink Throughput from End-user Measurements in Mobile Broadband Networks," in *2019 IEEE 20th Int. Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1–10. DOI: 10.1109/WoWMoM.2019.8792968.

[46] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP," *ACM SIGMOBILE mobile computing and communications review*, vol. 4, no. 3, pp. 47–57, 2000.

[47] H. Schulzrinne, "Personal mobility for multimedia services in the Internet," in *European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, Springer, 1996, pp. 143–161.

[48] Y.-j. Oh, E.-h. Paik, and K.-r. Park, "Design of a SIP-based real-time visitor communication and door control architecture using a home gateway," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1256–1260, 2006. DOI: 10.1109/TCE.2006.273142.

[49] T. Kim, H. Park, S. H. Hong, and Y. Chung, "Integrated system of face recognition and sound localization for a smart door phone," *IEEE Transactions on*

Consumer Electronics, vol. 59, no. 3, pp. 598–603, 2013. DOI: 10.1109/TCE.2013.6626244.

[50] S. A. Hadiwardoyo, "An overview of multicast routing techniques for group communications applications," in *2017 25th Telecommunication Forum (TELFOR)*, 2017, pp. 1–4. DOI: 10.1109/TELFOR.2017.8249423.

[51] W. Fenner, *RFC2236: Internet Group Management Protocol, Version 2*, IETF RFC 2236, USA, 1997.

[52] L. Costa and R. Vida, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*, IETF RFC 3810, Jun. 2004. DOI: 10.17487/RFC3810. [Online]. Available: https://www.rfc-editor.org/info/rfc3810.

[53] B. Fenner, M. J. Handley, I. Kouvelas, and H. Holbrook, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*, IETF RFC 4601, Aug. 2006. DOI: 10.17487/RFC4601. [Online]. Available: https://www.rfc-editor.org/info/rfc4601.

[54] K. K. Lam, J. M. Chapin, and V. W. Chan, "Performance analysis and optimization of multipath TCP," in *2011 IEEE Wireless Communications and Networking Conference*, 2011, pp. 695–700. DOI: 10.1109/WCNC.2011.5779217.

[55] M. Scharf and T.-R. Banniza, "MCTCP: A Multipath Transport Shim Layer," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1–5. DOI: 10.1109/GLOCOM.2011.6134021.

[56] M. J. Handley and V. Jacobson, *SDP: Session Description Protocol*, IETF RFC 2327, Apr. 1998. DOI: 10.17487/RFC2327. [Online]. Available: https://www.rfc-editor.org/info/rfc2327.

[57] G. Camarillo and J. Rosenberg, *The Alternative Network Address Types (ANAT) Semantics for the Session Description Protocol (SDP) Grouping Framework*, IETF RFC 4091, Jun. 2005. DOI: 10.17487/RFC4091. [Online]. Available: https://www.rfc-editor.org/info/rfc4091.

[58] T. R. N and R. Gupta, "A Survey on Machine Learning Approaches and Its Techniques:" in *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–6. DOI: 10.1109/SCEECS48394.2020.190.

[59] A. Montazerolghaem, M. H. Y. Moghaddam, and A. Leon-Garcia, "OpenSIP: Toward software-defined SIP networking," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 1, pp. 184–199, 2018, ISSN: 19324537. DOI: 10.1109/TNSM.2017.2741258.

[60] M. Kubat and J. Kubat, *An introduction to machine learning*. Springer, 2017, vol. 2.

[61] G. Rebala, A. Ravi, and S. Churiwala, *An introduction to machine learning*. Springer, 2019.

[62] S. Badillo, B. Banfai, F. Birzele, *et al.*, "An introduction to machine learning," *Clinical pharmacology & therapeutics*, 2020.

[63] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[64] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, *et al.*, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[65] J. Xie, F. R. Yu, T. Huang, *et al.*, "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2019. DOI: 10.1109/COMST.2018.2866942.

[66] L. Schmarje, M. Santarossa, S.-M. Schröder, and R. Koch, "A Survey on Semi-, Self- and Unsupervised Learning for Image Classification," *IEEE Access*, vol. 9, pp. 82 146–82 168, 2021. DOI: 10.1109/ACCESS.2021.3084358.

[67] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, *Unsupervised Learning of Visual Features by Contrasting Cluster Assignments*, 2021. [Online]. Available: https://arxiv.org/abs/2006.09882.

[68] Z. Chen, H. Zhang, W. G. Hatcher, J. Nguyen, and W. Yu, "A streaming-based network monitoring and threat detection system," in *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*, 2016, pp. 31–37. DOI: 10.1109/SERA.2016.7516125.

[69] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.

[70] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *arXiv preprint arXiv:2006.05278*, 2020.

[71] Z.-H. Zhou and Z.-H. Zhou, "Semi-supervised learning," *Machine Learning*, pp. 315–341, 2021.

[72] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[73] S. A. Fayaz, S. Jahangeer Sidiq, M. Zaman, and M. A. Butt, "Machine Learning: An Introduction to Reinforcement Learning," *Machine Learning and Data Science: Fundamentals and Applications*, pp. 1–22, 2022.

[74] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, 1996.

[75] Jürgen Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[76] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539.

[77] M. A. Nielsen, *Neural Networks and Deep Learning*, 2018. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[78] T. Mikolov, M. Karafiát, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model.," in *Interspeech*, Makuhari, 2010.

[79] C. L. Giles, G. M. Kuhn, and R. J. Williams, "Dynamic recurrent neural networks: Theory and applications," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 153–156, 1994. DOI: 10.1109/TNN.1994.8753425.

[80] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997. DOI: 10.1109/78.650093.

[81] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107 398, 2021, ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107398. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327020307846.

[82] L. O. Chua, *CNN: A paradigm for complexity*. World Scientific, 1998.

[83] A. D. Rasamoelina, F. Adjailia, and P. Sinčák, "A Review of Activation Function for Artificial Neural Network," in *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2020, pp. 281–286. DOI: 10.1109/SAMI48414.2020.9108717.

[84] D. Mitrović, M. Zeppelzauer, and C. Breiteneder, "Chapter 3 - Features for Content-Based Audio Retrieval," in *Advances in Computers: Improving the Web*, ser. Advances in Computers, Elsevier, 2010. DOI: https://doi.org/10.1016/S0065-2458(10)78003-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245810780037.

[85] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967. DOI: 10.1109/TAU.1967.1161901.

[86] L. Shi, I. Ahmad, Y. He, and K. Chang, "Hidden Markov model based drone sound recognition using MFCC technique in practical noisy environments," *Journal of Communications and Networks*, vol. 20, no. 5, pp. 509–518, 2018. DOI: 10.1109/JCN.2018.000075.

[87] W. Jun, "A speaker recognition system based on MFCC and SCHMM," in *Symposium on ICT and Energy Efficiency and Workshop on Information Theory and Security (CIICT 2012)*, 2012, pp. 88–92. DOI: 10.1049/cp.2012.1868.

[88] E. Larsen, C. Schmitz, C. Lansing, W. O'Brien, B. Wheeler, and A. Feng, "Acoustic scene analysis using estimated impulse responses," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 1, 2003, 725–729 Vol.1. DOI: 10.1109/ACSSC.2003.1292009.

[89] Y. Hioka, K. Niwa, S. Sakauchi, K. Furuya, and Y. Haneda, "Estimating Direct-to-Reverberant Energy Ratio Using D/R Spatial Correlation Matrix Model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2374–2384, 2011. DOI: 10.1109/TASL.2011.2134091.

[90] S. Vesa, "Binaural sound source distance learning in rooms," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 17, no. 8, pp. 1498–1507, 2009, ISSN: 15587916. DOI: 10.1109/TASL.2009.2022001.

[91] K. Zhagyparova, R. Zhagypar, A. Zollanvari, and M. T. Akhtar, "Supervised Learning-based Sound Source Distance Estimation Using Multivariate Features," *TENSYMP 2021 - 2021 IEEE Reg. 10 Symp.*, pp. 1–5, 2021. DOI: 10.1109/TENSYMP52854.2021.9551007.

[92] A. Brendel and W. Kellermann, "Learning-based acoustic source-microphone distance estimation using the coherent-to-diffuse power ratio," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2018-April, 2018. DOI: 10.1109/ICASSP.2018.8462474.

[93] Y. Li and H. Chen, "Reverberation Robust Feature Extraction for Sound Source Localization Using a Small-Sized Microphone Array," *IEEE Sensors Journal*, vol. 17, no. 19, pp. 6331–6339, 2017. DOI: 10.1109/JSEN.2017.2739144.

[94] P. Calamia, N. Balsam, and P. Robinson, "Blind estimation of the direct-to-reverberant ratio using a beta distribution fit to binaural coherence," *The Journal of the Acoustical Society of America*, vol. 148, no. 4, EL359–EL364, Oct. 2020, ISSN: 0001-4966. DOI: 10.1121/10.0002144. eprint: https://pubs.aip.org/asa/jasa/article-pdf/148/4/EL359/15347624/el359_1_online.pdf. [Online]. Available: https://doi.org/10.1121/10.0002144.

[95] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, 2015.

[96] S. Kobayashi, T. Hasegawa, T. Miyoshi, and M. Koshino, "MarNASNets: Toward CNN Model Architectures Specific to Sensor-Based Human Activity Recognition," *IEEE Sensors Journal*, vol. 23, no. 16, pp. 18 708–18 717, 2023. DOI: 10.1109/JSEN.2023.3292380.

[97] H. Cho and S. M. Yoon, "Divide and Conquer-Based 1D CNN Human Activity Recognition Using Test Data Sharpening," *Sensors*, vol. 18, no. 4, 2018, ISSN: 1424-8220. DOI: 10.3390/s18041055. [Online]. Available: https://www.mdpi.com/1424-8220/18/4/1055.

[98] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[99] X. Wang, M. Xia, and W. Deng, "MSRN-Informer: Time Series Prediction Model Based on Multi-Scale Residual Network," *IEEE Access*, vol. 11, pp. 65 059–65 065, 2023. DOI: 10.1109/ACCESS.2023.3289824.

[100] M. H. Rahman, M. A. S. Sejan, M. A. Aziz, Y.-H. You, and H.-K. Song, "HyDNN: A Hybrid Deep Learning Framework Based Multiuser Uplink Channel Estimation and Signal Detection for NOMA-OFDM System," *IEEE Access*, vol. 11, pp. 66 742–66 755, 2023. DOI: 10.1109/ACCESS.2023.3290217.

[101] A. Lawal, S. Rehman, L. M. Alhems, and M. M. Alam, "Wind Speed Prediction Using Hybrid 1D CNN and BLSTM Network," *IEEE Access*, vol. 9, pp. 156 672–156 679, 2021. DOI: 10.1109/ACCESS.2021.3129883.

[102] F. Shen, J. Liu, and K. Wu, "Multivariate Time Series Forecasting Based on Elastic Net and High-Order Fuzzy Cognitive Maps: A Case Study on Human Action Prediction Through EEG Signals," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 8, pp. 2336–2348, 2021. DOI: 10.1109/TFUZZ.2020.2998513.

[103] H. Kuttruff, *Room acoustics*. Crc Press, 2016.

[104] T. H. Falk, C. Zheng, and W.-Y. Chan, "A Non-Intrusive Quality and Intelligibility Measure of Reverberant and Dereverberated Speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1766–1774, 2010. DOI: 10.1109/TASL.2010.2052247.

[105] A. M.-C. So and Y. Ye, "On Solving Coverage Problems in a Wireless Sensor Network Using Voronoi Diagrams," in *Proceedings of the First International Conference on Internet and Network Economics*, ser. WINE'05, Hong Kong, China: Springer-Verlag, 2005, ISBN: 3540309004. DOI: 10.1007/11600930_58. [Online]. Available: https://doi.org/10.1007/11600930_58.

[106] T. R. N and R. Gupta, "A survey on machine learning approaches and its techniques:" in *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–6. DOI: 10.1109/SCEECS48394.2020.190.

[107] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 2027–2057, 2021. DOI: 10.1109/COMST.2021.3089688.

[108] P. Raut, H. Khandelwal, and G. Vyas, "A Comparative Study of Classification Algorithms for Link Prediction," in *2020 2nd International Conference on Inno-*

vative Mechanisms for Industry Applications (ICIMIA), 2020, pp. 479–483. DOI: 10.1109/ICIMIA48430.2020.9074840.

[109] K. R. Dalal, "Analysing the Role of Supervised and Unsupervised Machine Learning in IoT," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020, pp. 75–79. DOI: 10.1109/ICESC48915.2020.9155761.

[110] A. Cutler, D. Cutler, and J. Stevens, "Random Forests," *Machine Learning - ML*, Jan. 2011. DOI: 10.1007/978-1-4419-9326-7_5.

[111] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification.," in *GrC*, X. Hu, Q. Liu, A. Skowron, T. Y. Lin, R. R. Yager, and B. Zhang, Eds., IEEE, Mar. 22, 2007, ISBN: 0-7803-9017-2. [Online]. Available: http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/grc05.pdf.

[112] C. Zhang, J. Shi, L. Li, *et al.*, "Signaling Latency Analysis of Peer-to-Peer SIP Systems," in *2008 5th IEEE Consumer Communications and Networking Conference*, 2008, pp. 505–509. DOI: 10.1109/ccnc08.2007.118.

[113] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Commun. ACM*, Sep. 1975, ISSN: 0001-0782. DOI: 10.1145/361002.361007. [Online]. Available: https://doi.org/10.1145/361002.361007.

[114] *The Communications Market 2021*, Data from Ofcom, UK, Industry Insights, Jul. 2021. [Online]. Available: https://www.ofcom.org.uk/research-and-data/multi-sector-research/cmr/cmr-2021.

[115] H. Schulzrinne and J. Rosenberg, *Session Initiation Protocol (SIP): Locating SIP Servers*, IETF RFC 3263, Jul. 2002. DOI: 10.17487/RFC3263. [Online]. Available: https://www.rfc-editor.org/info/rfc3263.

[116] J. Costa-Requena, H. Tang, and I. E. Del Pozo, "SIP dealing with location based information," *Journal of Communications and Networks*, vol. 3, no. 4, pp. 351–360, 2001. DOI: 10.1109/JCN.2001.6596966.

[117] ITU-T, "One-way transmission time," International Telecommunication Union, Geneva, Recommendation G.114, May 2003.

[118] A. Brendel and W. Kellermann, "Learning-Based Acoustic Source-Microphone Distance Estimation Using the Coherent-to-Diffuse Power Ratio," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 61–65. DOI: 10.1109/ICASSP.2018.8462474.

[119] A. Ivry, B. Berdugo, and I. Cohen, "Voice Activity Detection for Transient Noisy Environment Based on Diffusion Nets," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 254–264, 2019. DOI: 10.1109/JSTSP.2019.2909472.

[120] G. Carter, C. Knapp, and A. Nuttall, "Estimation of the magnitude-squared coherence function via overlapped fast Fourier transform processing," *IEEE transactions on audio and electroacoustics*, vol. 21, no. 4, pp. 337–344, 1973.

[121] R. G. Brown, *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.

[122] E. S. Gardner Jr, "Exponential smoothing: The state of the art-Part II," *International journal of forecasting*, 2006.

[123] J. J. LaViola, "Double Exponential Smoothing: An Alternative to Kalman Filter-Based Predictive Tracking," in *Proceedings of the Workshop on Virtual Environments 2003*, ser. EGVE '03, Zurich, Switzerland: Association for Computing Machinery, 2003, ISBN: 1581136862. DOI: 10.1145/769953.769976. [Online]. Available: https://doi.org/10.1145/769953.769976.

[124] Box, George EP and Jenkins, Gwilym M and Reinsel, Gregory C and Ljung, Greta M, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[125] K. Chan and J. D. Cryer, *Time series analysis with applications in R*. Springer, 2008.

[126] X. Zou, S. Muramatsu, and H. Kiya, "The generalized overlap-add and overlap-save methods using discrete sine and cosine transforms for FIR filtering," in *Pro-*

ceedings of Third International Conference on Signal Processing (ICSP'96), vol. 1, 1996, 91–94 vol.1. DOI: 10.1109/ICSIGP.1996.566980.

[127] M. Slaney, "Auditory Toolbox: A MATLAB Toolbox for Auditory Modeling Work. Technical Report, version 2, Interval Research Corporation," Tech. Rep., 1998. [Online]. Available: https://engineering.purdue.edu/~malcolm/interval/1998-010/.

[128] L. Datta, *A Survey on Activation Functions and their relation with Xavier and He Normal Initialization*, 2020.

[129] V. N. Mitnala, M. J. Reed, I. Kegel, and J. Bicknell, "Seamless device handover for pervasive speech communication," in *2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, 2022, pp. 1–6. DOI: 10.1109/ICCSPA55860.2022.10018978.

[130] *Smart speakers research with the public*, Data from Ofcom, UK, Community Research, Dec. 2022. [Online]. Available: https://www.ofcom.org.uk/internet-based-services/smart-devices/smart-speaker-research/.

[131] Z. Chen, S. Zhang, S. McClean, *et al.*, "Process Mining IPTV Customer Eye Gaze Movement Using Discrete-Time Markov Chains," *Algorithms*, vol. 16, no. 2, 2023, ISSN: 1999-4893. DOI: 10.3390/a16020082. [Online]. Available: https://www.mdpi.com/1999-4893/16/2/82.

[132] M. L. Valero and E. A. P. Habets, "Multi-Microphone acoustic echo cancellation using relative echo transfer functions," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 229–233. DOI: 10.1109/WASPAA.2017.8170029.

[133] S. Ansari, K. A. Alnajjar, T. Khater, S. Mahmoud, and A. Hussain, "A robust hybrid neural network architecture for blind source separation of speech signals exploiting deep learning," *IEEE Access*, vol. 11, pp. 100 414–100 437, 2023. DOI: 10.1109/ACCESS.2023.3313972.

[134] K. Nagatomo, M. Yasuda, K. Yatabe, S. Saito, and Y. Oikawa, "On-line sound event localization and detection for real-time recognition of surrounding environment," *Applied Acoustics*, vol. 199, p. 108 961, 2022, ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2022.108961. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X22003358.