Analysing and Evaluating the Coexistence Strategies of IPv4 and IPv6 in the Real-World Network.

(Submitted on 25th April 2024)

Student Name: Sang Min Lee

Supervisor Name: Wanqing Tu

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract — This project investigates the coexistence methods of IPv4 and IPv6 and evaluates the effectiveness of each approach. The study involves an extensive collection of network traffic data from various Internet Service Providers (ISPs), captured using Wireshark based on meticulously designed experimental scenarios. Additionally, custom software development was undertaken to facilitate the analysis and evaluation of the captured data. This comprehensive analysis aims to understand how ISPs manage the transition from IPv4 to IPv6, highlighting the operational challenges and performance outcomes of different coexistence strategies. The project provides insights into the future of network evolution and recommendations to enhance the networking environment in the face of growing internet demands.

Index Terms—IPv4, IPv6, Coexisting method, Dual-stack, Tunneling, Teredo, RTT, Jitter, Throughput, KT, BT, Vodafone, Virgin Media, ISP, Networking, Wireshark, Mobile Internet Service Provider, Protocol, UDP, TCP.

1 Introduction

1.1 Project Description

These days, numerous devices are being deployed and they are connected, the proliferation of networking devices has reached unprecedented levels, each requiring its unique IP address for communication. However, the finite range of IPv4 addresses has become increasingly strained as the demand continues to surge. This exhaustion of available IPv4 addresses poses a critical challenge to the function of the internet. Recognising this crisis, IPv6 addressing was developed as a solution to the limitations of IPv4. Therefore, IPv6 provides a large address space, ensuring the growth and scalability of the internet continuously. Although IPv6 has benefits, the deployment of IPv6 has been interrupted by a variety of technical and financial barriers. The obstacles to the transition to IPv6 are from various factors. The existing systems and infrastructure that have been built around IPv4 require significant investment in technical and financial factors to upgrade or replace, presenting a formidable financial obstacle for numerous organisations. Additionally, concerns in terms of compatibility, interoperability, and the complexity of implementation further, disrupt the switching IPv6. However, IPv4 still

has a limited number of spaces, therefore; it should be replaced by IPv6. As the number of available IPv4 addresses continues to decrease, there has been an increased reliance on temporary solutions such as NAT, and IPv4 and IPv6 coexistence methods have also been widely deployed, and are now being used by various ISPs.
Therefore, the project concentrates on a comprehensive investigation into the practical implementation and efficacy of IPv4 and IPv6 coexisting methods, with a particular focus on capturing and analysing network traffic data from specific Internet Service Providers (ISPs). Deploying advanced network analysis tools such as Wireshark, the purpose of the project is to gain insights into the real-world internet of IPv4 and IPv6 coexistence within different network environments. Therefore, various packet and traffic data have been captured, also using Wireshark. It is based on individual experimental scenarios and also captured data from four major Internet Service Providers (ISPs), BT, Vodafone, and Virgin Media in the United Kingdom, and KT in South Korea. Moreover, there is a code implementation that helps to analyse and evaluate the packet data. Then, the specific evaluation methodologies were applied to analyse and evaluate the performance. Also, occupancy of each coexistence method, Identification of the coexistence method and the performance of each ISP's coexistence method have been conducted.

1.2 Project Timeline

The Gantt chart has been displayed in Appendix 1.
The chart shown in Appendix 1 clearly describes the official deadline for each task of the project. In the chart, individual internal deadlines and plans are delivered. The first task, Literature Survey and Project Plan was submitted on 26th October. Also, between 16th October and 26th October, network contents were reviewed. From 21st November to 7th December, the preparation for the first presentation was conducted as the actual presentation date was on 8th December. Before the Christmas holiday, data was supposed to be captured in the U.K., however; due to the difficulties in finding appropriate ISPs, it was delayed. Then, during the Christmas holiday, the data collection in Korea was performed. After the holiday, collected data started being analysed. After the analysis of the collected data, the code implementation has begun. Progress demonstration was delayed due to personal medical reasons. After the demonstration session, the final project paper and the preparation for the oral exam began.

1.3 Deliverables

There are 3 types of deliverables, basic, intermediate, and advanced. that are involved in the project.

a. **Basic deliverables**
   Understandings of IPv4 and IPv6 addressing: the overview of IPv4 and IPv6 addressing will be contained in a later section.
   The designed experimental scenario: it is used for packet capture, also showing specific aspects to make the experiment more efficient.
   The overview of each coexistence method: the specific IPv4 and IPv6 coexisting methods are listed. Those methods are the investigated ISPs are currently applying. This deliverable also contains an explanation of each method.
b. **Intermediate deliverables**
   Visual Documentation: it is provided for the section, Data Analysis and Results. It will be shown with some graphs and values of each measurement.
   Occupancy of each coexistence method & Identification of coexistence method: analyses how selected ISPs manage and transition between IPv4 and IPv6 protocols using specific coexisting methods.
   In-depth coexisting methods analysis: each captured data is analysed in detail with appropriate measurements, such as RTT, Jitter and Throughput. In the analysis, the strengths and weaknesses are explained with proper references.
   Software Development: the software is provided, which helps to analyse and evaluate the captured data easily. The implementation identifies coexisting methods which specific ISPs are currently deploying, also evaluating the performance with proper measurements.
c. **Advanced Deliverables**
   Achieved Results: This is relevant to the analysis of coexisting methods above. Also, the outcome of this aspect is with some graphs and interpretations of the analysis.
   Research Paper: the paper summarises the entire project's findings, insights, methodology, evaluation, result and conclusion.

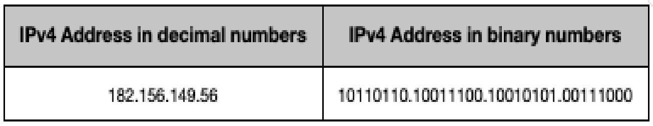
2 RELATED WORK

This section contains the background knowledge for the project. The understanding of IPv4 and IPv6 can be found in this section. Also, the contents of basic IPv4 and IPv6 coexistence methods and their strengths and weaknesses are discussed.

2.1 The understanding of IPv4 addressing

Internet Protocol (IP) addresses serve as the cornerstone of Internet communication, uniquely identifying each connected device within the Transmission Control Protocol/Internet Protocol (TCP/IP) framework. IPv4, the fourth iteration of the Internet Protocol, is characterised by several key features that underpin its addressing scheme.

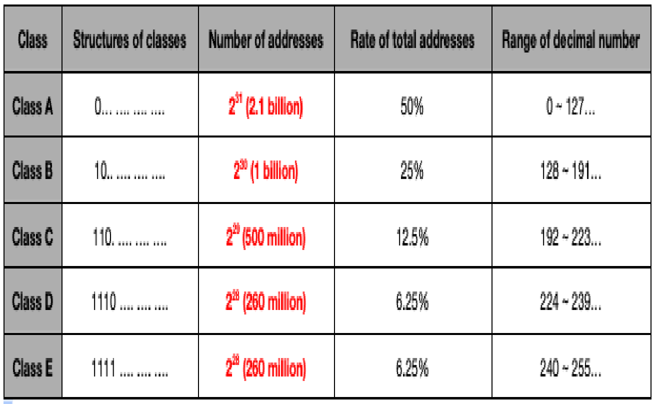
TABLE 1
EXAMPLES OF IPV4 ADDRESSING



IPv4 Address in decimal numbers	IPv4 Address in binary numbers
182.156.149.56	10110110.10011100.10010101.00111000

Hung Ngo [1] introduced that IPv4 addresses are structured as 32-bit address systems, comprising two primary components: the network prefix and the host number. Within the given network, every the host shares the same network address while possessing a unique host number, enabling individual device identification. Arctic Guru [2] wrote that IPv4 addresses can be either locally or globally unique, depending on the scope of network visibility. Devices restricted to internal networks utilise locally unique addresses, while those requiring external connectivity, such as websites and online games, employ globally unique addresses. Traditionally, IPv4 addresses are represented in decimal notation, facilitating human readability. However, underneath this representation lies a binary foundation, with each IPv4 address consisting of 32 binary digits. Consequently, IPv4 addresses are inherently finite, with a total address space of 2^32, covering a range from 0.0.0.0 to 255.255.255.255. In order to manage IPv4 addresses efficiently, the concept of 'classful addressing' was introduced, although it was later superseded by 'classless addressing'. Classful addressing organises IPv4 addresses into five distinct classes, ranging from Class A to Class E, based on address range and allocation policies. However, the rigid structure of classful addressing resulted in inefficient address utilisation and hindered scalability, prompting the adoption of classless addressing principles.

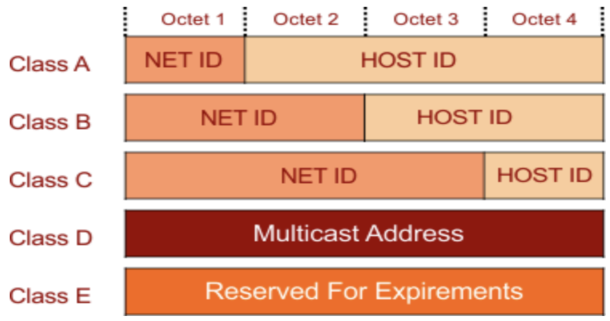
TABLE 2
CLASSLESS ADDRESSING



Class	Structures of classes	Number of addresses	Rate of total addresses	Range of decimal number
Class A	0...	2^{31} (2.1 billion)	50%	0 ~ 127...
Class B	10..	2^{30} (1 billion)	25%	128 ~ 191...
Class C	110.	2^{29} (500 million)	12.5%	192 ~ 223...
Class D	1110	2^{28} (260 million)	6.25%	224 ~ 239...
Class E	1111	2^{28} (260 million)	6.25%	240 ~ 255...

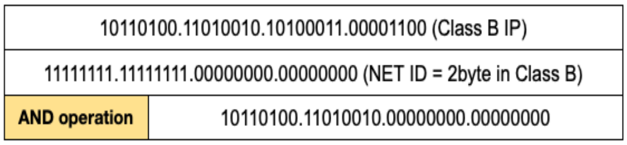
Fuller V and Li T [3] specified that classless addressing, epitomised by Classless Interdomain Routing (CIDR), revolutionised IP address allocation by abandoning fixed class boundaries in favour of flexible address aggregation. CIDR enables the dynamic allocation of IP address blocks based on specific routing criteria, facilitating more efficient address utilisation and better accommodating the evolving needs of the internet.

TABLE 3
CLASSFUL ADDRESSING



Subnetting, a fundamental aspect of classless addressing, involves partitioning a large address block into smaller, contiguous sub-blocks to enhance network efficiency and address space utilisation. Subnets function as encapsulated networks within a larger network structure, optimising routing paths and minimising address wastage. By converting host ID bits into network ID bits, subnetting allows for the creation of subnetworks tailored to diverse organisational requirements.

TABLE 4
SUBNETTING



10110100.11010010.10100011.00001100 (Class B IP)	
11111111.11111111.00000000.00000000 (NET ID = 2byte in Class B)	
AND operation	10110100.11010010.00000000.00000000

In addition, subnetting reduces IP address wastage by allowing the creation of subnets with different host capacities, thereby optimising address allocation and network resource utilisation. Fuller V and Li T [3] assigned that by specifying subnet masks in CIDR notation (e.g. "192.168.1.1/28"), subnetting delineates network boundaries and determines the size of subnets within a given address block. In summary, the transition from class-based to classless addressing, accompanied by the introduction of subnetting and CIDR, represents a major evolution in IPv4 networking. These advances not only address the challenges of address exhaustion and inefficient allocation but also pave the way for a more scalable, resilient and adaptive Internet infrastructure.

2.1 The understanding of IPv6 addressing

Addressing is where the most notable disparities between IP version 4 (IPv4) and IPv6 can be found, primarily focusing on how addresses are structured and utilised. The Knowledge Academy [4] claimed that IPv6 boasts a significantly larger address space compared to the increasingly depleted IPv4 counterpart. This is accomplished by expanding the IP address size from the 32-bit characteristic of IPv4 to a spacious 128-bit. Each additional bit allocated to an address effectively doubles the overall address space.

Juniper [5] introduced that IPv4 has had to be augmented through methods such as Network Address Translation (NAT), which consolidates ranges of private addresses into a single public address, along with temporary address assignment. While these approaches have their merits, they are insufficient for accommodating the demands of newer applications and environments like emerging wireless technologies, always-on connectivity, and Internet-connected consumer devices.

In addition to the augmented address space, IPv6 diverges from IPv4 addresses in several key ways, including the introduction of a scope field that designates the address's intended application, the omission of support for broadcast addresses in favour of multicast addresses for packet broadcasting, and the introduction of a novel address type called anycast.

IPv6 builds upon the functionality and structure of IPv4 by providing a simplified and enhanced packet header to allow for more efficient routing and improving support for mobile phones and other mobile computing devices. IPv6 addresses consist of 128 bits, instead of 32 bits, and include a scope field that identifies the type of application suitable for the address. IPv6 does not support broadcast addresses, but instead uses multicast addresses for broadcast. In addition, IPv6 defines a new type of address called anycast.

Unicast addresses, for instance, are tailored to support global address scope as well as two distinct types of local address scope. The first type, known as link-local unicast addresses, is exclusively employed within a single network link. These addresses are identifiable by the first 10 bits of the prefix and cannot be utilised beyond the boundaries of that particular network link. The second type, site-local unicast addresses, serves the purpose of communication solely within a specific site or intranet, which may encompass multiple network links. These addresses are indicative of nodes within the intranet and are restricted for use within the site, with no capability for external usage.

On the other hand, multicast addresses provide a wide array of address scope options, numbering 16 in total, which includes node, link, site, organisation, and global scope. To distinguish between these different scopes, a 4-bit field within the prefix is used to identify the address scope.

A single interface is identified by unicast addresses, with each unicast address composed of a prefix containing 'n' bits and an interface ID consisting of the remaining 128 - n bits. On the other hand, multicast addresses specify a group of interfaces. Each multicast address comprises an initial 8 bits of all '1s', a 4-bit flags section, a 4-bit scope designation, and a 112-bit group ID.

TABLE 5
SUBNETTING

```
11111111 | flgs | scop | group ID
```

The first octet, filled with '1s', is used to indicate that the address is multicast. Within this address, the flags field is responsible for determining whether the multicast address is known or transient, while the scope field specifies the range or domain of the multicast address. The 112-bit group ID within the address uniquely identifies the particular multicast group.

Similarly to multicast addresses, anycast addresses target a group of interfaces, but the difference is that packets are delivered to only one of these interfaces, rather than all of them. Anycast addresses are taken from the standard unicast address space and have the same format as unicast addresses, making it difficult to distinguish them by structure alone. Consequently, each member of an anycast group must be configured to recognise certain addresses as anycast addresses.

IPv6 addresses are 128 bits which are typically written in the format consisting of 8 segments of 16 bits. These segments are represented in hexadecimal notation, ranging from 0 to FFFF. Colons are deployed as delimiters between segments, and any leading zeros in any segment are ignored. If there are two or more consecutive segments including zeros only, those zeros can be combined into a double colon (::). IPv6 addresses are structured as 8 sets of 16-bit hexadecimal values separated by colons (:). The general format of IPv6 addresses is as follows.

TABLE 6
IPV6-1

```
aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa
```

Each "aaaa" represents a 16-bit hexadecimal value, while each "a" denotes a 4-bit hexadecimal value.

TABLE 7
IPV6-2

```
3FFE:0000:0000:0001:0200:F8FF:FE75:50DF
```

There is the option to eliminate the leading zeros in each 16-bit group, as demonstrated below.

TABLE 8
IPV6-3

```
3FFE:0:0:1:200:F8FF:FE75:50DF
```

16-bit groups of zeros can be compressed to double colons (::) as shown in the image below, however; it can only be compressed once per address.

TABLE 9
IPV6-4

```
3FFE::1:200:F8FF:FE75:50DF
```

An IPv6 address prefix is formed by combining an IPv6 prefix (address) and a prefix length. The prefix is represented in the format ipv6-prefix/prefix-length and designates a range of address space or a network. The ipv6-prefix adheres to standard IPv6 addressing conventions. The prefix length is a numeric value in decimal, specifying the count of consecutive, higher-order bits in the address that constitute the network portion. For instance, 10FA:6604:8136:6502::/64 is an example of an IPv6 prefix with zero compression. In this address, the site prefix is found in the leftmost 64 bits, which are 10FA:6604:8136:6502.

2.3 The Introduction of Coexistence Methods

This section involves the introduction of the project-related IPv4 and IPv6 coexistence methods. To begin with, the Dual-stack method is one of the most straightforward coexistence methods. Steffann S, Colitti L, and Gashinsky I. [6] introduced that it also contains running both IPv4 and IPv6 simultaneously on the same network devices. Additionally, it allows devices to communicate over either protocol, addressing the gap between former and modern network environments. Steffann S, Colitti L, and Gashinsky I. [6] also introduce that Tunnelling is a coexistence method deployed to encapsulate IPv6 traffic within IPv4 packets. It enables the transmission of IPv6 data over an IPv4 network without requiring any changes to the existing IPv4 devices. It also enables the phased introduction of IPv6 by utilising the existing network pathways.
Teredo Tunneling is a specific type of tunnelling that facilitates the transmission of IPv6 packets across IPv4 networks, containing those configurations that deploy Network Address Translation (NAT). It dynamically assigns IPv6 addresses to IPv4 nodes, enabling them to participate in the IPv6 internet. Additionally, it is especially beneficial for clients behind NAT devices to access the IPv6 internet.

2.4 Strengths and Weaknesses

Applying IPv4 and IPv6 Coexistence Methods has both benefits and weaknesses. For benefits, it enhances flexibility, compatibility and risk mitigation. Where coexistence methods are deployed, IPv6 can be easily adopted without any changes in network infrastructures and environments. It also enables interoperability between IPv4 and IPv6 to ensure that all users, regardless of the version of the IP protocol on the network, can continue to access the service. Furthermore, both protocols can be tested and used together and it allows reducing the risk associated with the transition to new technologies. However, it still has several concerns related to complexity, cost and delay in full adoption. Managing the two IP protocols increases the complexity of network design, operation, and maintenance. Additionally, dual configuration, staff training, and increased operational costs often require additional resources. The availability of transitional methods can reduce the urgency for full deployment of IPv6 and extend the transition period.

3 METHODOLOGY

As the project primarily contains the analysis of specified collected traffic data based on individual experimental scenarios and code implementation, this section introduces those aspects.

3.1 Experimental Scenario

This section describes the individual experimental scenario and considerations. The basic logic of the experimental scenario is simulating the relevant applications in terms of each type of data.

a. Text

For text data, online messengers and emails are used. specifically, Whatsapp, KaKaoTalk, and Microsoft Outlook.

1. Use a device for the relevant applications to capture traffic in the network interface using Wireshark.
2. Analyse the captured packets to observe the messaging protocol (e.g., XMPP)
3. Analyse captured packets to observe how IPv4 and IPv6 coexist during the communication.

b. Audio

For audio data, VoIP calls are conducted using related applications such as WhatsApp, KaKaoTalk and Microsoft Teams. Also, music streaming applications are deployed such as Spotify and YouTube music.

1. Initiate a VoIP call via the devices executing WhatsApp, KaKaoTalk and Microsoft Teams. Play the music on Spotify and YouTube music.
2. Capture traffic within the network using Wireshark.
3. Focus on the SIP, RTP and UDP protocols for audio data.
4. Analyse captured packets to evaluate the coexistence of IPv4 and IPv6 during the VoIP call.

c. Video

For video data, video streaming and video calls are simulated through the applications YouTube, Microsoft Teams, WhatsApp, KaKaoTalk and Zoom.

1. Stream videos using the applications above to capture network traffic on the relevant interface using Wireshark.
2. Focusing on UDP and QUIC protocols during packet capture.
3. Analyse captured packets to assess how IPv4 and IPv6 coexist during video streaming.

3.2 Considerations during the experiment

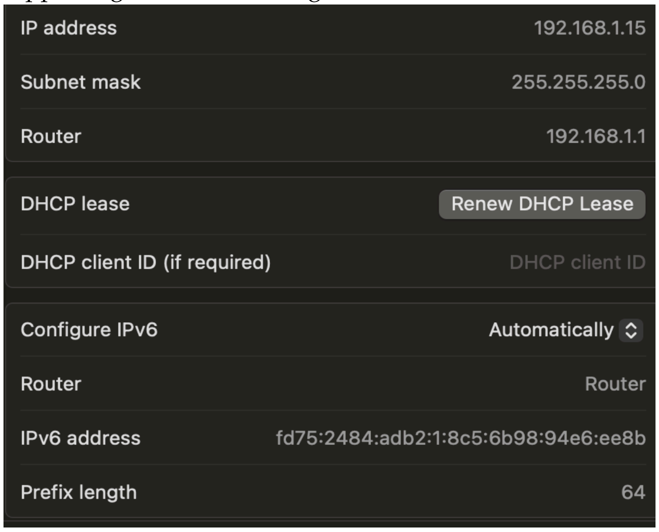
Before the capture of various traffic, the source IP should be identified and relevant protocols must be considered. Additionally, for security reasons, some applications use their protocols and traffic data is encrypted. Furthermore, for the identification of the protocols, it is essential to identify the specific protocols used by each application before capturing traffic. This involves not only recognising standard protocols such as SIP, RTP, UDP, and QUIC but also detecting any application-specific protocols or variations. Understanding these protocols will aid in accurately filtering and analysing the data relevant to IPv4 and IPv6 coexistence. The majority of modern applications encrypt their traffic to protect user data. Therefore, it is important to recognise the limitations this may impose on data analysis, as encrypted packets may not reveal detailed protocol information. Where possible, consider using simulation tools or developer modes within the applications to gain access to unencrypted data flow. In order to handle unexpected scenarios, it is important to develop contingency plans for potential issues such as network downtime, data corruption, or lack of IPv6 support, as encountered with Korean ISPs. Additionally, technical setup is significant to ensure that all devices used in the experiment are correctly configured for both IPv4 and IPv6 traffic. This includes network interface settings, firewall configurations, and any relevant routing protocols that might impact data capture.

3.3 Data Collection

This project involves the collection of network traffic data from three major British ISPs, BT, Vodafone, and Virgin Media as well as a mobile internet service provider from Korea. The data collection efforts are framed by the experimental scenarios outlined previously, which simulate the real-world usage of various data-intensive applications under IPv4 and IPv6 coexistence environments.

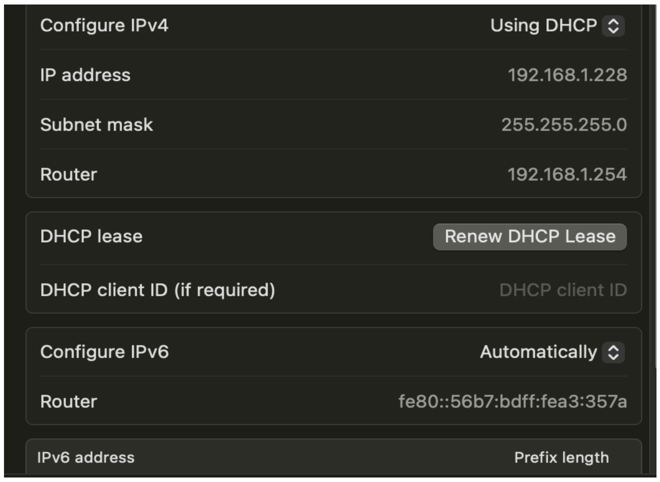
3.3.1 Preparation and Initial Setup

Before initiating data capture, local IP addresses for each ISP were verified to confirm the deployment of IPv6. This step is crucial to ensure that the data collected is relevant to our study on IPv4 and IPv6 coexistence. The network readiness for IPv6 was confirmed visually and documented in figures, as referenced earlier, to demonstrate that each ISP's network is capable of supporting IPv6 traffic alongside IPv4.



IP address	192.168.1.15
Subnet mask	255.255.255.0
Router	192.168.1.1
DHCP lease	Renew DHCP Lease
DHCP client ID (if required)	DHCP client ID
Configure IPv6	Automatically
Router	Router
IPv6 address	fd75:2484:adb2:1:8c5:6b98:94e6:ee8b
Prefix length	64

Fig 3.1. The local IP address of Vodafone



Configure IPv4: Using DHCP
IP address: 192.168.1.228
Subnet mask: 255.255.255.0
Router: 192.168.1.254
DHCP lease: Renew DHCP Lease
DHCP client ID (if required): DHCP client ID
Configure IPv6: Automatically
Router: fe80::56b7:bdff:fea3:357a
IPv6 address | Prefix length

Fig 3.2. The local IP address of BT

Various applications were executed to generate and analyse traffic typical of everyday internet usage. This includes data-intensive services such as online messaging, VoIP calls, video streaming, and other relevant applications that utilise both IPv4 and IPv6 protocols.

Each application was chosen based on its popularity and typical usage scenarios to reflect a realistic environment.

3.3.2 Traffic capture

Wireshark was utilised to capture and log the network traffic. This tool is instrumental in providing detailed insights into the packet-level activity on a network. Specific filters were applied to isolate traffic by protocol type, which is essential for detailed analysis of performance metrics such as jitter, RTT, throughput, and packet loss under IPv4 and IPv6 coexistence. Also, filters were meticulously set up to target packets relevant to the project, ensuring that the captured data is precisely focused on the coexistence of IPv4 and IPv6, as well as the specific protocols used by the applications under the project.

This approach allows for an accurate measurement of how each ISP handles IPv4 and IPv6 traffic and their deployed coexistence methods. Following data capture, the next steps involve detailed analysis using both quantitative and qualitative methods to assess the network performance of IPv4 and IPv6 coexisting methods. This structured approach ensures a comprehensive evaluation of each ISP's performance across different coexistence methods.

3.4 Data Analysis

3.4.1 Coexistence methods applied in the project

There are many reasons why Internet Service Providers (ISPs) deploy IPv4 and IPv6 coexistence methods without fully transitioning to IPv6, despite the clear long-term benefits of IPv6, which are multi-faceted and based on practical, economic, and technical realities. Firstly, the ubiquity of IPv4 can be considered. IPv4 has been around since the 1980s, and countless systems, devices, and services have been built to use it. This widespread adoption means that switching from IPv4 is not just a matter of flipping a switch, but a major overhaul of the existing infrastructure. Also, Community FS [7] stated that there are compatibility issues because not all existing devices and services support IPv6 yet. Many older systems and internet-connected devices (such as certain IoT devices) only recognise IPv4. For ISPs, it is also important to ensure that all customers can access all parts of the internet, regardless of their device's capabilities, therefore, keeping IPv4 alongside IPv6 is necessary for full compatibility. ISPs still take into account the gradual transition and economic factors. The Internet is a vast network of many entities (such as businesses, governments, and individuals), and coordinating a complete transition to IPv6 is extremely complex. ISPs that deploy both IPv4 and IPv6 can minimise disruption by making a gradual transition that allows them to operate and test both protocols simultaneously. As well as, Completely replacing IPv4 infrastructure with IPv6-compatible technology involves a significant financial investment, which is prohibitively expensive for many ISPs, especially in economically under-resourced regions. Therefore, running both protocols allows ISPs to invest in IPv6 incrementally as their budgets allow, rather than all at once. In order to address the exhaustion of IPv4 addresses, ISPs deployed NAT (Network Address Translation) which has extended the life of IPv4, making the immediate transition less urgent for some ISPs. Although NAT helps to manage IPv4 addresses, IPv4 is still not enough to cover all existing devices. Therefore, ISPs are currently deploying coexistence methods. There are 3 representative IPv4 and IPv6 coexistence methods in the project.

A. Tunneling

Tunneling is one of the representative IPv4 and IPv6 coexistence methods, allowing IPv4 and IPv6 networks to coexist and communicate with each other. Tunnelling is similar to a tunnel that allows two network environments to connect seamlessly, even if they use different IP versions. To explain the process of Tunneling, there are 3 important stages. The first step is encapsulation. During the encapsulation process, S. Konyeha, E. Osa, E. Evbuomwan [8] introduced that IPv6 packets are wrapped into IPv4 packets. For example, for postage, letters are normally put into the envelopes. Here, the letters are IPv6 and the envelopes are IPv4. After encapsulation, encapsulated packets are sent over an IPv4 network. This part is equivalent to sending an envelope through the postal system, which only understands how to handle the envelope and does not understand the letters inside the envelope. After that, the decapsulation is executed. For instance, when a packet arrives at the destination where IPv6 is understood, the IPv4 'envelope' is removed and IPv6 'Letters' is read.

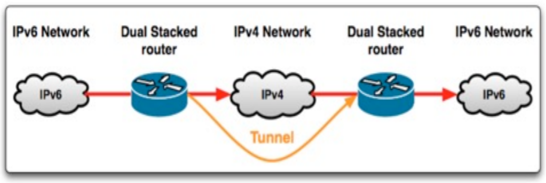


Fig 3.3. Tunnelling (source from https://blog.apnic.net/2015/12/21/test-your-ipv6-via-tunnelling/)

As you can see in the figure above, those two networks use an IPv4 infrastructure and they are

IPv6-based networks. The tunnel allows two networks to communicate with each other. Also, when IPv6 address packets pass through the tunnel, the packets are encapsulated within IPv4 addresses.

B. Dual-Stack

The dual-stack belongs to the IPv4 and IPv6 coexistence methods. This method involves the simultaneous operation of both IPv4 and IPv6 on the same network infrastructure. Additionally, it allows the devices to fully support and fully operate in both IPv4 and IPv6 networking environments. Furthermore, it provides a robust strategy for transitioning between IPv4 and IPv6. To introduce how Dual-stack is implemented, the first factor is the network configuration. In a Dual-Stack network, both IPv4 and IPv6 devices such as routers, switches and end devices can be configured and operated. Shrivastava K, Shukla S, Tiwari S [9] claimed that the devices have both IPv4 and IPv6 addresses, as well as; they can send and receive traffic using either protocol.

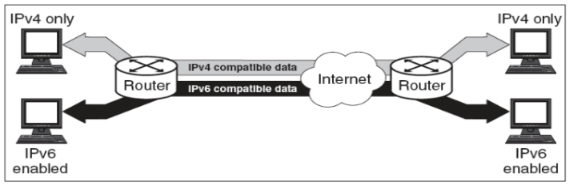


Fig 3.4. Dual-Stack (source from https://www.cables-solutions.com/what-is-ipv4-ipv6-dual-stack-and-mpls-technique.html)

Referring to the figure above, Martín JE, García J, Gomez D, et al [10] specified that the Dual-Stack method allows the devices to communicate with other devices even though the source devices and the destination devices use different IP versions, for example, from IPv4 to IPv6 and vice versa. The protocol selection within the transmitting can be allocated automatically, however; it is frequently based on the availability of the destination address and DNS resolution. In terms of DNS configuration, it is similar to the IP configuration. DNS servers in the Dual-Stack network handle both A and AAAA records, the former one is for IPv4 and the latter one is for IPv6. When the devices request domain names, they can receive both record types and decide the protocol for the connection.

C. Teredo Tunneling

Teredo Tunneling is widely used for the NAT environment where direct IPv6 connections are blocked or unavailable. Telecom Trainer [11] introduced that Teredo Tunnelling enables IPv6 connection behind the NAT which is based on the IPv4 environment.

A Teredo client which is typically a host requiring IPv6 connectivity interacts with a Teredo Server which helps to initially conduct the tunnel.

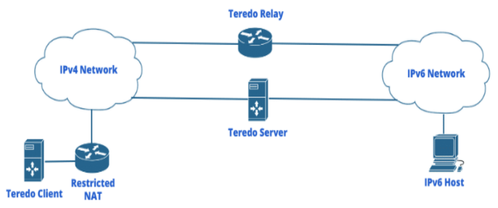


Fig 3.5. Teredo Tunnlling (source from https://www.rapidseedbox.com/blog/teredo)

As it belongs to Tunneling, it also encapsulates packets. Primarily, IPv6 packets are encapsulated within the UDP packets of IPv4, which allows the IPv6 packets to be sent over the IPv4 network as if they were supposed to be regular IPv4 UDP traffic. Huitema C and Krčo S [12] also claimed that the encapsulated packets can be passed through NAT devices which are configured by IPv4. For the tunnel establishment, the client sends a packet to the Teredo server to identify the type of NAT devices. Also, through this process, it acquires a public IPv4 address. When the server responds, it assists the client in establishing its address and path for tunnelling IPv6 traffic. Once the tunnel is prepared for communication, IPv6 packets can be sent from the client. Also, Teredo relies on relay routers which allow the transfers of encapsulated packets between IPv4 networks and IPv6 networks.

3.4.2 The measurements

This section describes the methodologies employed to analyse network performance in scenarios where IPv4 and IPv6 coexist, focusing on UDP traffic. UDP is a connectionless protocol that does not guarantee packet delivery, making it crucial for applications requiring fast, real-time communication such as streaming and gaming. The performance metrics evaluated are jitter, round-trip time (RTT), throughput, and packet loss, as measured from UDP packets captured with Wireshark.

a. Jitter

Liang G and Yu FR [13] argued that Jitter is the variation in time between packet arrivals, which is created by either network congestion, timing drift, or route change. High jitter would lead to a poor user experience in real-time applications. The formula of jitter is shown below.

Jitter_i = ABS((D_(i-1, i)) - (D_(i, i+1))), where $d_{i-1,i}$ *and* $d_{i,i+1}$ *are the delays between consecutive packets.*

b. RTT

Fulber-Garcia V [14] introduced that RTT measures how long it takes for a signal to be transmitted and how long it takes for an acknowledgement of that signal to be received from the source again, which is critical to understanding the latency of the network. The formula of RTT is shown below.

RTT = (Time of ACK received) - (Time of Packet Sent)

This formula applies to situations where UDP packets trigger a response from the server, such as receiving ICMP messages in response to a sent packet.

c. Throughput
Saxena A [15] specified that Throughput is the rate at which messages are successfully delivered over a communication channel. It is particularly important for assessing the capacity of the network to handle data. Higher throughput values indicate better utilisation of available network resources and improved data transfer rates. The formula of Throughput is shown below.
Throughput = (Total Bits Received) / (Total Time in Seconds)

d. packet loss
Packet loss represents the percentage of packets that are sent but do not successfully reach their destination. This can occur due to errors in data transmission, network congestion, or route changes. The formula of Throughput is shown below.
*Packet Loss = ((Total Packets Sent - Total Packets Received) / Total Packets Sent) * 100%*

3.5 Code Implementation

Primarily, Python has been deployed for the implementation. Also, it helps to analyse network packet captures, specifically focusing on coexistence methods between IPv4 and IPv6 and detailed network performance metrics for communication between specific IP addresses. It is based on the Pyshark library for packet parsing, which abstracts much of the complexity of handling Pcap files directly. Visualisation has been delivered based on the Matplotlib library. So, the implementation provides a graphical representation of the data, enhancing the understanding of network behaviour.
The implementation has initialisation which allows the implementation to start by loading a capture file. Additionally, it interacts with the users by displaying options in terms of the main features and functions of the implementation.
load_capture_file() prompts the user to enter the path to a Wireshark capture file (.pcapng) and uses Pyshark to load this capture file into a Pyshark FileCapture object. It also requests the file path from the user. Then, it initialises a Pyshark FileCapture object with this file path, which loads the capture file for analysis.
identify_coexistence_method(cap) determines the coexistence method of IPv4 and IPv6 within the loaded capture file and also counts occurrences of IPv4 and IPv6 packets, specifically counts IPv6 packets with link-local addresses (fe80:: prefix). It also decides the network coexistence strategy (e.g., Dual Stack, 6to4 Tunneling, etc.) based on the content and proportion of these addresses and returns a string indicating the identified coexistence method. Specifically, for Dual-stack, it checks if the packet is a DNS packet containing an AAAA record. If such a record is found, it sets the *has_aaaa_record* flag to True. Then, it determines it is a Dual-stack. Additionally, for Teredo Tunneling, if the prefix is *2001:*, it is detected as Teredo Tunneling.
analyze_ipv6_usage(cap) analyses the usage of IPv6 packets within the capture and provides a visual summary. It counts total frames and IPv6 packets and calculates the ratio of IPv6 packets to total frames. Then, it prints total frames, IPv6 packet count, and the IPv6 usage ratio. It also uses the Matplotlib library to plot these statistics visually. The most significant and complicated function in the implementation is *calculate_metrics(cap, src_ip, dst_ip, analyze_ipv4, analyze_ipv6)* to calculate and display network metrics like Round Trip Time (RTT), Jitter, Throughput, and Packet Loss between specified source and destination IPs for either IPv4 or IPv6 protocols. (see the appendix 2)
First, RTT is calculated as the difference between the time when a packet is sent and the time when the ACK (acknowledgement) is received. This formula is applied in actual implementation,
rtt = info['received'] - info['sent'][0]['timestamp'].
Here, *info['received']* is the timestamp when a response (or next related packet) is received, and
info['sent'][0]['timestamp'] is the timestamp of the initial packet sent.
Also, there is a specific calculator for Jitter which can be calculated using the formula, *jitter = sum(abs(info['sent'][i]['timestamp'] - info['sent'][i-1]['timestamp'] - rtt) for i in range(1, len(info['sent'])))*. It calculates the absolute difference between the arrival time of consecutive packets and the average RTT and then sums these differences. This method highlights variations in delay and is used to gauge the stability of network latency.
Also, the implementation supports acquiring throughput value using the formula, *total_bytes_sent = sum(packet['length'] for packet in info['sent'])*
*throughput = (total_bytes_sent * 8) / (rtt / 1000000)*
To interpret the formula above, *total_bytes_sent* is the sum of the sizes of all sent packets. Then, it multiplies by 8 and converts bytes to bits. This formula assumes that throughput should be measured over the round trip time of the first packet sent to the last acknowledgement received.
Additionally, in the implementation, the packet capture object *(cap)* is reused across different functions, which involves iterating over the capture multiple times. It is important to reset the capture iterator using *cap.reset()* before re-iterating over the packets, especially after a full iteration has been completed in one function and another function is about to start a new iteration. If this is not handled properly, the second function might not process any packets because the iterator is already at the end of the capture. The exception handling (*try...except*) has been implemented which manages potential runtime errors, such as issues with loading the capture file or processing packet data.
Furthermore, the use of Matplotlib for visualisation in the *analyze_ipv6_usage* function was crucial for providing a visual representation of the data analysis. This enhances understanding and allows for quick identification of data trends or anomalies. The structure of the implementation is also modular, separating the functionalities into distinct functions. Therefore, this enhances readability and maintainability. Especially, the functions, *identify_coexistence_method*, *analyze_ipv6_usage*, and *calculate_metrics* are self-contained, each responsible for specific tasks. This modular design facilitates easier updates and modifications to individual aspects of the implementation without impacting other parts. Moreover, its main loop allows for continuous interaction until the

user decides to exit. This interactive mode enhances user experience by enabling real-time data analysis based on user input without needing to restart the implementation. In the *calculate_metrics* function, the use of dictionaries to store packet information keyed by tuple identifiers (e.g., *(src_ip, dst_ip, src_port, dst_port)*) is an efficient way to organise and retrieve packet data for analysis. This method supports complex analyses such as RTT, Jitter, and Throughput calculations by maintaining detailed transaction records.

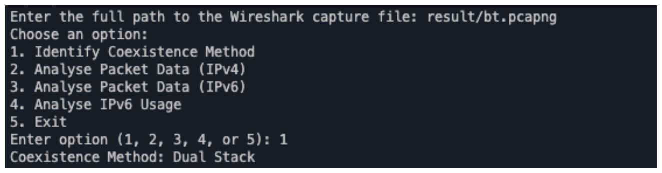
4 Results

This section presents the findings from the analysis of network traffic data collected from three major British ISPs BT, Vodafone, and Virgin Media and a Korean mobile internet service provider, KT. The data capture involved various applications and services running under both IPv4 and IPv6 environments to assess the real-world implementation and efficiency of different coexisting methods. The results are divided into two main subsections below.

4.1 Occupancy of each coexistence method & Identification of coexistence method

This section examines the specific IPv4 and IPv6 coexisting methods employed by selected Internet Service Providers (ISPs), focusing on their strategies to manage and transition between these two protocols. British ISPs, BT and Vodafone, have implemented the Dual Stack approach, while KT Mobile, a major Korean service provider, utilises Teredo Tunneling. These methods are integral to understanding how ISPs balance widespread IPv4 infrastructure with the adoption of IPv6.

A. BT



```
Enter the full path to the Wireshark capture file: result/bt.pcapng
Choose an option:
1. Identify Coexistence Method
2. Analyse Packet Data (IPv4)
3. Analyse Packet Data (IPv6)
4. Analyse IPv6 Usage
5. Exit
Enter option (1, 2, 3, 4, or 5): 1
Coexistence Method: Dual Stack
```

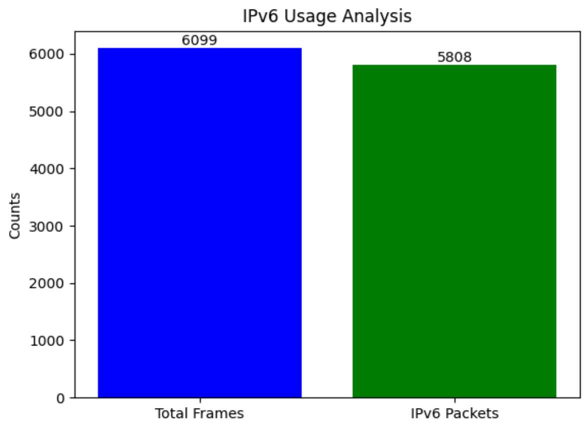
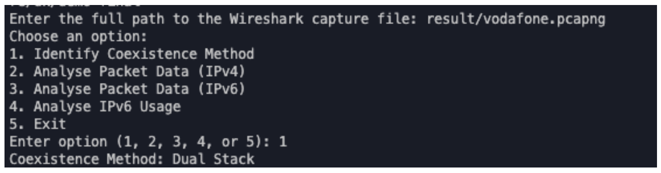


Fig 4.1. Analysis of BT

As shown in the figure above, BT is currently deploying the Dual-stack as their coexistence method. The total frames of the captured file are 6099, and 5808 IPv6 packets are occupied which is 95.23% of the total frames.

B. Vodafone



```
Enter the full path to the Wireshark capture file: result/vodafone.pcapng
Choose an option:
1. Identify Coexistence Method
2. Analyse Packet Data (IPv4)
3. Analyse Packet Data (IPv6)
4. Analyse IPv6 Usage
5. Exit
Enter option (1, 2, 3, 4, or 5): 1
Coexistence Method: Dual Stack
```

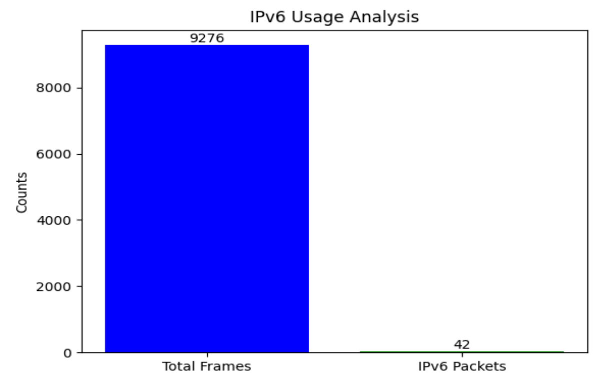


Fig 4.2. Analysis of Vodafone

According to the figure above which represent the captured data of Vodafone, which is also applying Dual-stack as their coexistence method. Additionally, the number of total frames is 9276. Only 42 packets are IPv6 packets, which is 0.45% of the total number of frames.

C. Virgin Media

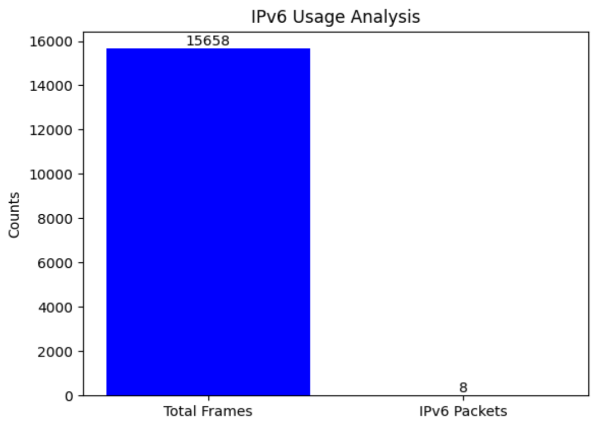
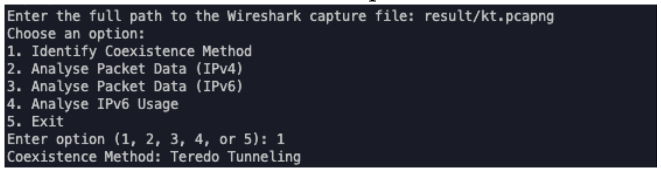


Fig 4.3. Analysis of Virgin Media

As shown in the figure above, 15658 frames have been captured in the Virgin Media network. There are only 8 IPv6 packets which occupy 0.05% of total frames.

D. KT - Korean mobile internet provider



```
Enter the full path to the Wireshark capture file: result/kt.pcapng
Choose an option:
1. Identify Coexistence Method
2. Analyse Packet Data (IPv4)
3. Analyse Packet Data (IPv6)
4. Analyse IPv6 Usage
5. Exit
Enter option (1, 2, 3, 4, or 5): 1
Coexistence Method: Teredo Tunneling
```

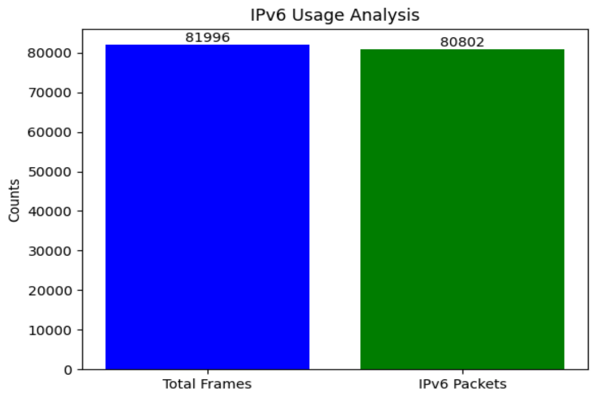


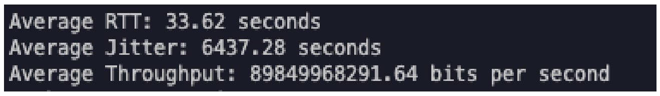
Fig 4.4. Analysis of KT

The figure above are from the packet data in the KT network. KT is currently deploying Teredo Tunneling as their coexistence method. The total number of captured frames is 81996, and the total number of IPv6 packets is 80802, accounting for 98.54% of the total captured frames.

4.2 The performance of each ISP's coexistence method

The performance evaluation of IPv4 and IPv6 coexisting methods offers critical insights into the efficiency and effectiveness of network protocols deployed by the analysed ISPs. This section presents a comprehensive analysis of key performance metrics jitter, RTT, throughput, and packet loss across each coexisting method, providing valuable insights into their impact on network performance and user experience.
Graphs and the result of each calculation are delivered in the Appendix section, please see the Appendix 3- 6.

A. BT

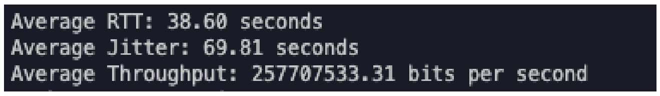


```
Average RTT: 33.62 seconds
Average Jitter: 6437.28 seconds
Average Throughput: 89849968291.64 bits per second
```

Fig 4.5. The performance of BT

The average value of RTT is 33.62 seconds, and the average value of Jitter is 6437.28 seconds. Also, the average value of throughput is 89849968291.64 bits per second. As well as a packet loss has been captured.
The relevant visualisation is shown in Appendix 3.

B. Vodafone

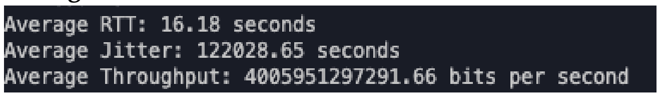


```
Average RTT: 38.60 seconds
Average Jitter: 69.81 seconds
Average Throughput: 257707533.31 bits per second
```

Fig 4.6. The performance of Vodafone

In the captured Vodafone data, the average RTT value is 38.60 seconds, and the average Jitter value is 69.81 seconds. Additionally, the average throughput value is 257707533.31 bits per second. Any packet losses have not been discovered.
The relevant visualisation is shown in Appendix 4.

C. Virgin Media

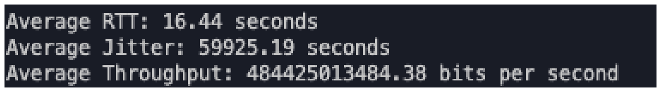


```
Average RTT: 16.18 seconds
Average Jitter: 122028.65 seconds
Average Throughput: 4005951297291.66 bits per second
```

Fig 4.7. The performance of Virgin Media

For the result of Virgin Media data, the average value of RTT is 16.18 seconds while the average Jitter value is 122028.65 seconds. Moreover, the average throughput value is 4005951297291.66 bits per second. Any packet losses have not been discovered.
The relevant visualisation is shown in Appendix 5.

D. KT



```
Average RTT: 16.44 seconds
Average Jitter: 59925.19 seconds
Average Throughput: 484425013484.38 bits per second
```

Fig 4.8. The performance of KT

To examine the performance evaluation of captured KT packet data, its RTT average value is 16.44 seconds while its Jitter average value is 59925.19 seconds. Furthermore, its average throughput value is 484425013484.38 bits per second. A packet loss has been captured.
The relevant visualisation is shown in Appendix 6.

4.3 Overall evaluation of each ISP's performance

BT and KT show a strong IPv6 adoption rate, which sets them apart as leaders in preparing for future internet protocol standards. BT's approach with Dual Stack and KT's with Teredo Tunneling provide robust models for successful IPv6 integration.
For Vodafone, it shows extremely low IPv6 occupancy rates although having Dual Stack technology in place. This indicates a need for the ISP to not only promote IPv6 more aggressively among their user bases but also to reassess their network policies or customer equipment to facilitate greater IPv6 usage. Vodafone and Virgin Media may be able to strategise effective roll-outs or awareness campaigns to increase IPv6 integration, ensuring network readiness for future demands.
Additionally, the majority of ISPs show high RTT and jitter values that far exceed typical expectations for modern networks, where RTTs are generally less than a second and jitter should not exceed a few milliseconds for optimal performance. These metrics suggest either a data reporting error or severe network configuration or capacity issues. High throughput in certain cases indicates decent data handling capacity but does not compensate for the poor latency and variability metrics that affect the quality of service. The absence of packet loss in Vodafone and Virgin Media is positive, however; the presence of it in BT and KT, combined with other poor metrics, raises concerns about network stability and reliability. Recommendations for ISPs would involve investigating and addressing the causes of high latency and jitter, potentially upgrading hardware or optimising network configurations to improve performance consistency and reduce delays.

4.3 Description of the applied evaluation method

For IPv4 and IPv6 packet identification, packets were categorised by protocol version to analyse the occupancy rates of IPv4 and IPv6. In terms of performance metrics, key performance indicators such as Jitter, RTT, Throughput, and Packet Loss were extracted for each session. Metrics were processed using statistical software to compute average values and detect unexpected circumstances. This includes calculating the mean for

RTT, Jitter, and Throughput, and counting instances of Packet Loss. The proportion of IPv6 versus IPv4 packets was calculated to assess the degree of IPv6 adoption and the effectiveness of the coexistence method. Additionally, graphs and charts were generated using the Python library, Matplotlib to visually represent the data. This facilitated easier interpretation and comparison of the ISPs' performance metrics.

5 Evaluation

5.1 Overall strengths of the project

The results section provides a decent insight into the IPv4 and IPv6 coexistence methods across different ISPs. This section also clearly divides the analysis into IPv6 adoption and performance metrics, providing a detailed look at both the occupancy of IPv6 and its performance implications across different ISPs. By examining both the extent of IPv6 adoption and the performance implications, the analysis highlights critical differences in how ISPs are transitioning to IPv6. Collecting and analysing real-world data through Wireshark ensures that the findings are practical in actual ISP behaviour rather than theoretical models. I believe that this practical insight is useful for real-world application and strategic planning. The direct comparison between ISPs regarding IPv6 adoption and performance metrics allows for an understanding of the competitive landscape.
Additionally, by presenting quantitative data on IPv6 occupancy and performance metrics such as RTT, jitter, throughput, and packet loss, the section offers concrete evidence to support conclusions in terms of the efficacy of different coexistence methods. Dividing the results into subsections for each ISP allows for direct comparison and easier digestion of how each network handles IPv6 integration and performance challenges.
The code implementation is well-structured into distinct functions, each handling specific tasks (loading data, identifying coexistence methods, analysing IPv6 usage, and calculating performance metrics). This makes the code easier to maintain and extend. The implementation also allows users to choose what analysis to perform and tailors the tool to varied user needs and scenarios. Moreover, it deploys a command-line interface for user interaction, allowing users to select specific analyses. This flexibility enhances the usability of the tool in different scenarios. It also includes visualisations such as bar charts for IPv6 usage to help make the data analysis more accessible and understandable to users. It integrates a library, Matplotlib for data visualisation directly in the analysis method allowing for immediate visual understanding of complex data sets. The implementation has been designed based on the modular design, with separate functions for different tasks, following good programming practices, facilitating easier updates, debugging, and scalability of the codebase.

5.2 Overall limitations of the project

In the project, there are some of the reported values, especially jitter, are exceptionally high and might indicate potential errors in evaluation or analysis. This could mislead interpretations unless validated. The project also potentially has the issue of the lack of a comparative baseline without a baseline for comparison. Therefore, it is challenging to definitively assess whether the reported figures represent decent, average, or poor performance. There is also potential data skew. The occupancy and performance evaluations are highly dependent on the quality and extent of the data captured. The results might not be generalisable across all scenarios or times. Moreover, network conditions can vary widely due to external factors not controlled during data capture, such as network congestion, and maintenance activities during the data collection period. While the results are detailed, there might be a lack of transparency regarding the specific conditions under which data was collected. Details such as the time of day, the duration of data capture, and the types of applications generating the traffic would be critical to fully interpret the results.
For the code implementation, the current implementation iterates over the packets multiple times for different functions which can be computationally expensive, especially with large datasets.There is minimal error handling in the code, particularly when dealing with unexpected or malformed data inputs, which could lead to crashes or incorrect outputs. The implementation keeps all packet data in memory, which can be inefficient and limit the size of the datasets that can be analysed on machines with limited memory. It could be addressed by implementing more memory-efficient handling of packet data, possibly by processing data in chunks or using more efficient data structures, which would improve performance.

5.3 Potential Mitigation of the Limitations

Algorithms can be implemented to automatically detect and exclude outliers or abnormal data that might skew the results, particularly in metrics such as jitter and throughput. The data collection could be conducted over various times of the day, different days of the week, and various network conditions to capture a broader range of scenarios. It may lead to making the results more generalisable across typical network behaviours.
For code implementation, refactor the code to process data in a single pass, storing intermediate results in a structured format such as a database or a dictionary that other functions can access. Therefore, it may reduce redundancy and improve processing time. more efficient data structures and algorithms can be implemented that can handle large datasets with lower memory overhead. Robust validation for all user inputs can be applied to prevent errors due to incorrect data entries, ensuring the implementation behaves predictably even with incorrect or unexpected inputs.
For large datasets, data streaming or the use of databases can be applied to manage data efficiently without loading everything into memory.

6 Conclusions

This conclusion refines the findings from the project on IPv4 and IPv6 coexistence methods across ISPs including BT, Vodafone, Virgin Media, and KT. The analysis focuses on how these providers handle the transition from IPv4 to IPv6, addressing global internet connectivity challenges and IPv4 limitations.
The methodology of the project involved data collection via Wireshark, evaluating the impact of various coexistence methods. Results demonstrated that while UK ISPs primarily implement Dual-stack methods, KT in Korea utilises Teredo Tunneling for IPv6 connectivity over IPv4 networks. The conclusion summarises these findings, highlights strengths such as detailed data analysis, addresses limitations, and proposes directions for future research.

6.1 Summary of Contents

In the introduction and related work sections, the project began with a detailed exploration of the operational characteristics and limitations of IPv4 and the enhanced capabilities of IPv6.
The methodology section described the process of capturing and analysing network traffic data across various applications and services. This section contains various subsections, Experimental Scenario, Considerations during the experiment, Data Collection, Data Analysis and Code Implementation.
The results section highlighted significant differences in coexistence methods among the ISPs. British ISPs primarily utilised Dual-stack methods, allowing simultaneous support for IPv4 and IPv6, whereas KT deployed Teredo Tunneling to facilitate IPv6 connectivity across IPv4 networks. The analysis also delved into performance metrics such as RTT, jitter, throughput, and packet loss, providing insights into the operational impact of each coexistence strategy.
The evaluation section emphasised the strengths of the study, including the depth of data analysis and broad coverage of ISP strategies, while also acknowledging limitations such as potential data anomalies and the geographical scope of the ISPs studied. Recommendations for future research were made, suggesting broader data collection and advanced analytical techniques.

6.2 Analysis of Findings

The comprehensive analysis conducted in this project has yielded specific insights into the IPv4 and IPv6 coexistence methods of major ISPs such as BT, Vodafone, Virgin Media, and KT. The findings from the study highlight the diverse approaches ISPs adopt to facilitate the transition to IPv6, driven by the exhaustion of IPv4 addresses and the need for enhanced network capabilities.
The findings revealed that BT, Vodafone, and Virgin Media predominantly employ the Dual-stack method. This approach allows these ISPs to operate IPv4 and IPv6 simultaneously, ensuring compatibility and continuity of service during the transition phase. The Dual-stack method has been shown to be effective in the maintenance of service reliability and user experience without the demand for significant investment in existing infrastructure. Contrarily, KT(Korea Telecom) has implemented Teredo Tunneling, which encapsulates IPv6 packets within IPv4 UDP packets. This method is also beneficial in environments where direct IPv6 connectivity is not available, such as behind NAT devices. The project also focused on key performance metrics such as Round-Trip Time (RTT), jitter, throughput, and packet loss, which are crutial for assessing the quality of network service. While Dual-stack methods generally exhibited stable performance metrics, Teredo Tunneling showed variability in throughput and higher RTT, reflecting the complexity and overhead of tunneling mechanisms. These findings suggest that while Dual-stack is suitable for ISPs with modern infrastructure, Teredo Tunneling offers a viable alternative for areas with limited IPv6 support. The diverse coexistence strategies analysed in this project emphasises the challenges and solutions in the transition towards IPv6.

6.3 Future Insights and Research Directions

Future studies should aim to extend the period and diversity of data collection. Long-term data collection across different network conditions and more varied geographical locations would provide a richer dataset for analysing trends and variability in network performance under IPv6 coexistence strategies.
As technologies such as 5G, and edge computing continue to proliferate, understanding their impact on IPv4 and IPv6 coexistence becomes crucial. Future research should explore how these technologies influence network requirements and coexistence strategies, particularly in terms of security, scalability, and performance. Further research could also explore the impact of national and international policies on the adoption rates of IPv6. Analysing the economic factors of the IPv6 transition may lead to further insights into the costs and benefits associated with different transition strategies. Those studies may help ISPs and other stakeholders make informed investment decisions regarding network upgrades and technology deployments.

REFERENCES

[1] Hung Ngo, *Understanding IP Addressing: Everything You Ever Wanted To Know,* https://cse.buffalo.edu/~hungngo/classes/2010/589/reading-materials/IP-addressing.pdf

[2] Arctic Guru, *Understanding IPv4 Address Structure,* https://arcticguru.com/ipv4-address-structure

[3] Fuller V, Li T, *RFC 4632 - Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan,* https://datatracker.ietf.org/doc/html/rfc4632

[4] The Knowledge Academy, *"Difference Between IPv4 and IPv6: What's the Difference?",* https://www.theknowledgeacademy.com

[5] Juniper, *IPv4 and IPv6 Protocol families,* https://www.juniper.net/documentation/us/en/software/junos/interfaces-security-devices/topics/topic-map/security-interface-ipv4-ipv6-protocol.html

[6] Steffann S, Colitti L, Gashinsky I, *RFC 7059: A Comparison of IPv6-over-IPv4 Tunnel Mechanisms,* IETF. RFC

[7] Community FS, *How to Achieve IPv4 and IPv6 Coexistence: Dual Stack or MPLS Tunnel?,* https://community.fs.com/blog/how-to-achieve-ipv4-and-ipv6-coexistence-dual-stack-or-mpls-tunnel.html

[8] S. Konyeha, E. Osa, E. Evbuomwan, *Demonstrating the Impact of 6to4 Tunneling on IPV4 and IPV6 Network Coexistence,* European Journal of Electrical Engineering and Computer Science, Vol. 4, No. 5.

[9] Shrivastava K, Shukla S, Tiwari S, *Dual-Stack IPv4 and IPv6 Coexistence and Transition: A Study. Journal of Scientific and Engineering Research.* Unpublished.

[10] Martín JE, García J, Gomez D, et al, *IPv6, IPv4, and Dual-Stack Lite: Analysis and Implementation. Journal of Network Systems Management.* Unpublished.

[11] Telecom Trainer, *Teredo Tunneling IPv6 over UDP through NATs,* https://www.telecomtrainer.com/teredo-tunneling-ipv6-over-udp-through-nats/

[12] Huitema C, Krčo S, Teredo, *IPv6 over UDP beyond NATs.* Unpublished.

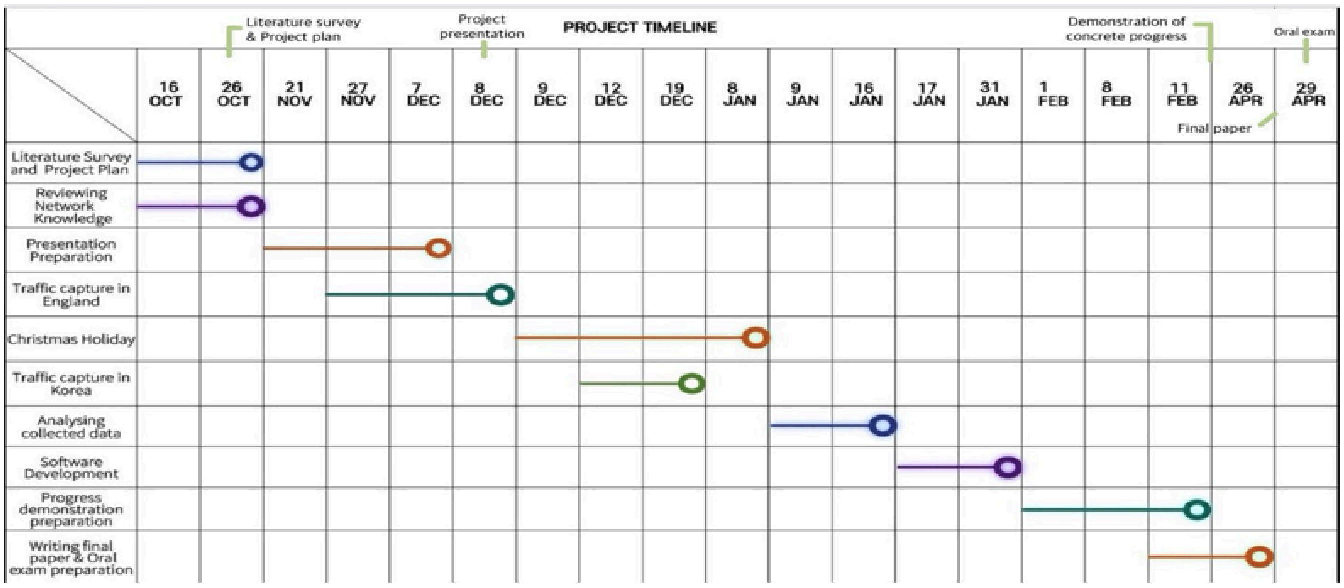
[13] Liang G, Yu FR, *Real-time communication in packet-switched networks,* IEEE.

[14] Fulber-Garcia V, *What Is Round Trip Time? Baeldung on Computer Science,* https://www.baeldung.com/cs/networking-round-trip-time.

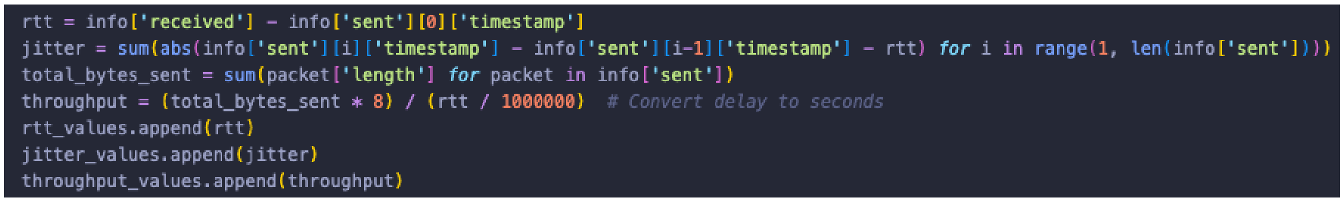
[15] Saxena A, T*hroughput in Computer Networks,* https://www.scaler.com/topics/throughput-in-computer-networks/

Appendix

1. Project time line (Gantt chart)

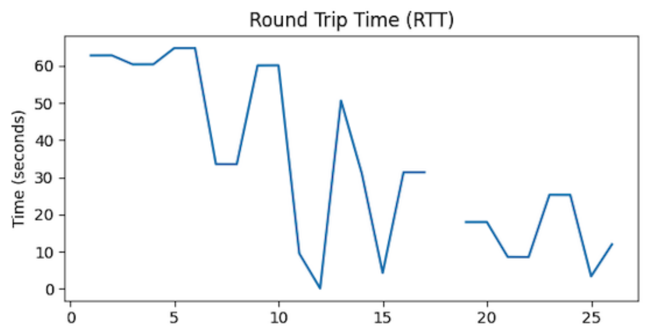


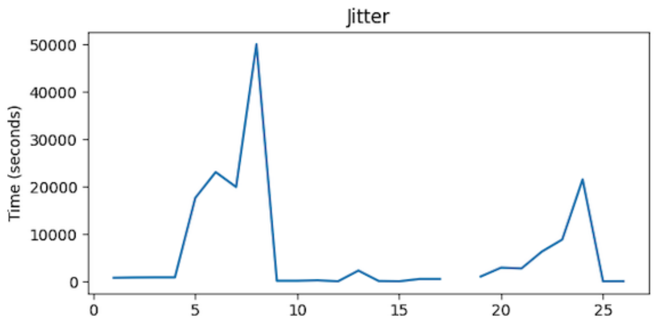
2. Formulas of rtt, jitter, throughput

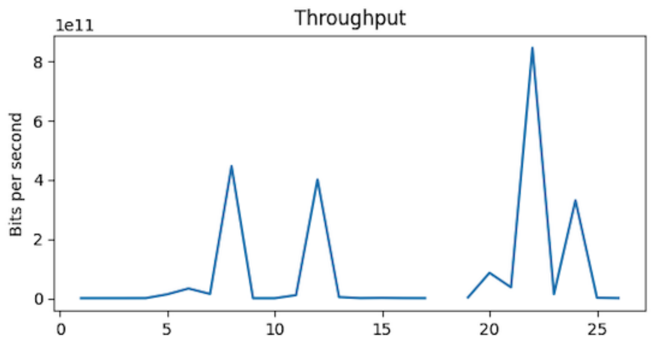


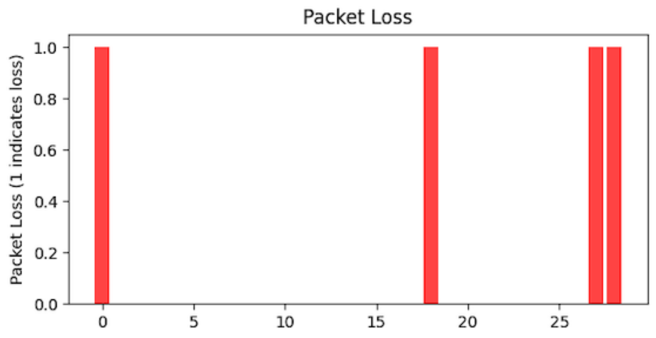
```
rtt = info['received'] - info['sent'][0]['timestamp']
jitter = sum(abs(info['sent'][i]['timestamp'] - info['sent'][i-1]['timestamp'] - rtt) for i in range(1, len(info['sent'])))
total_bytes_sent = sum(packet['length'] for packet in info['sent'])
throughput = (total_bytes_sent * 8) / (rtt / 1000000)  # Convert delay to seconds
rtt_values.append(rtt)
jitter_values.append(jitter)
throughput_values.append(throughput)
```

3. Performance metrics of BT

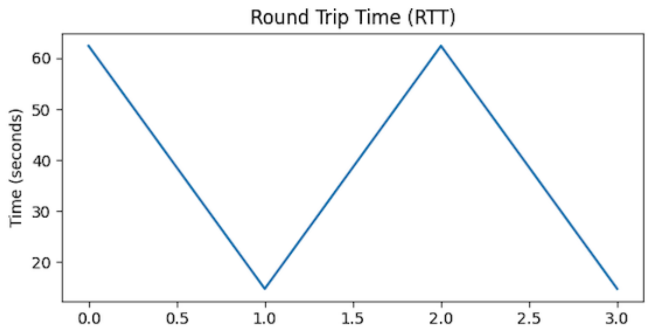


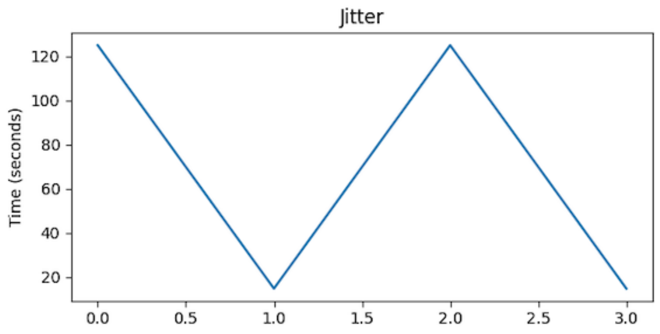


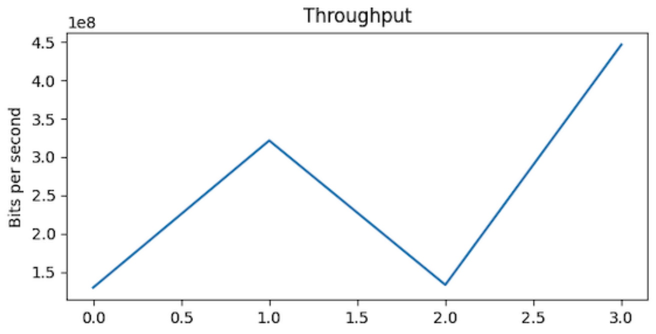




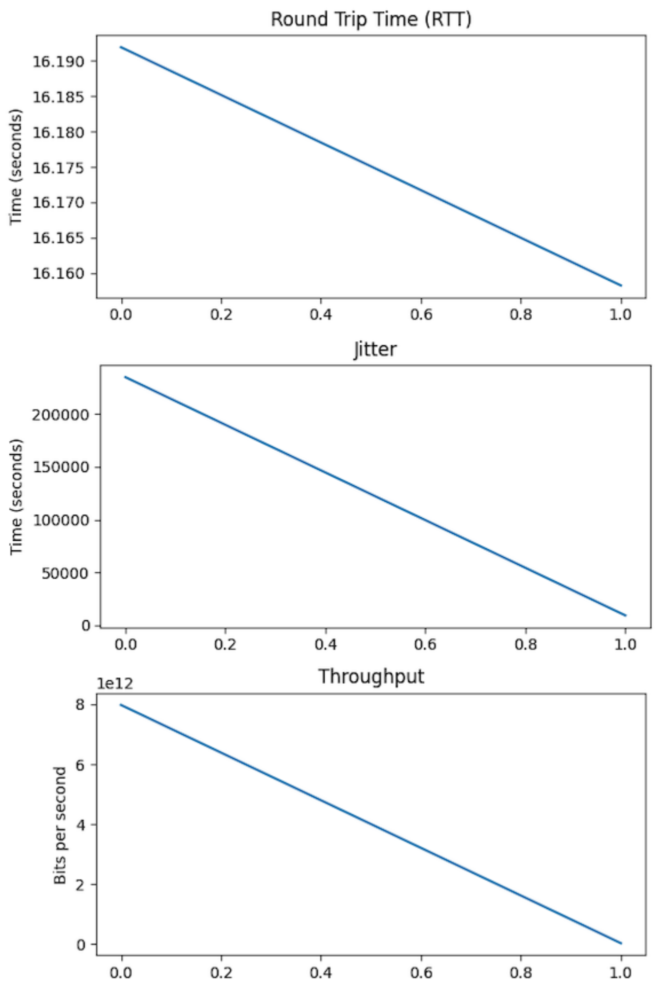
4. PERFORMANCE METRICS OF VODAFONE



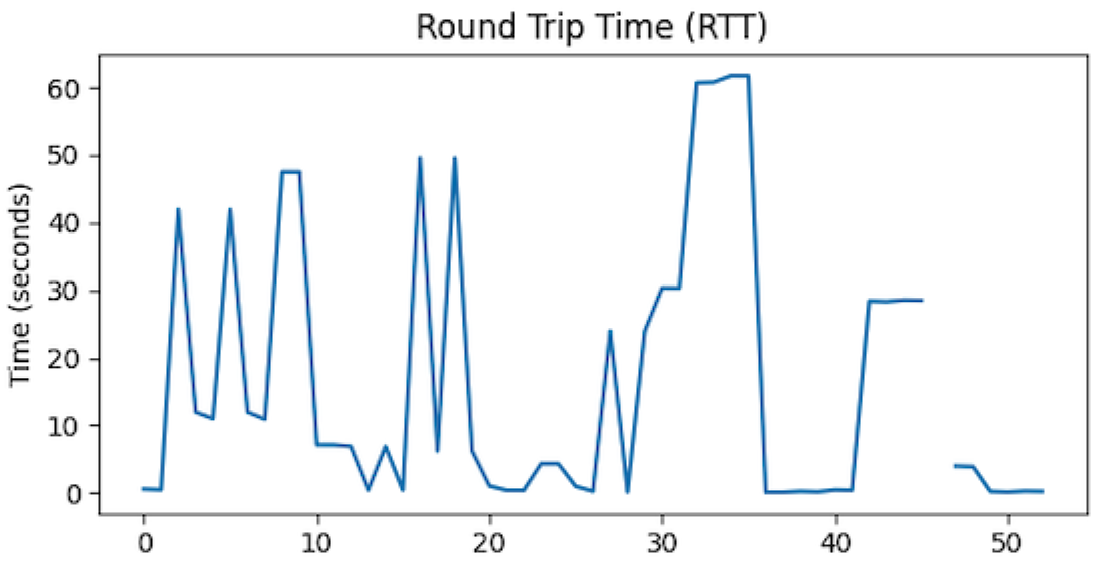


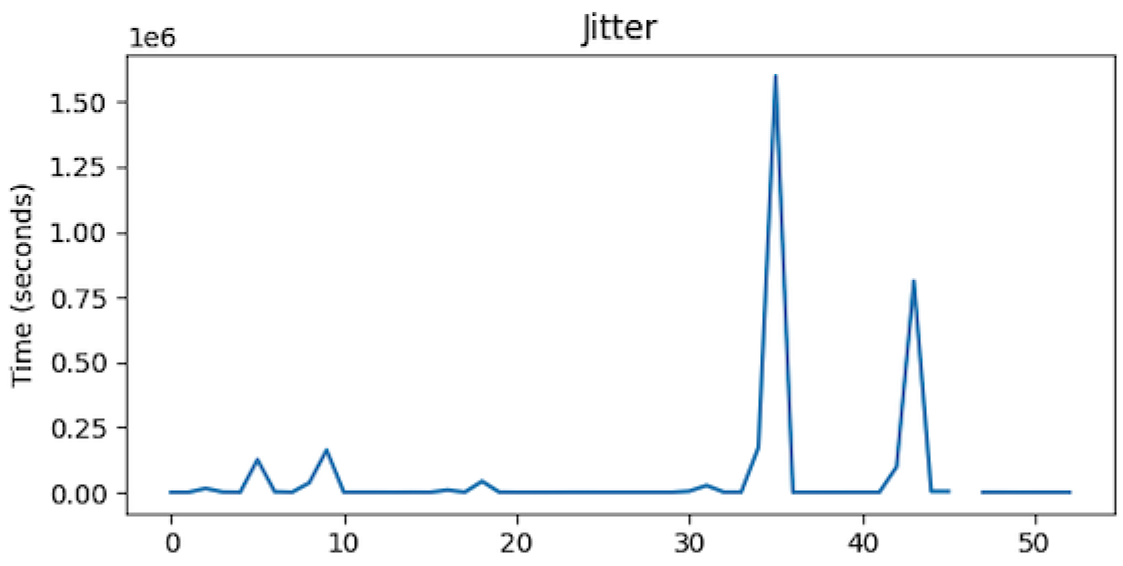


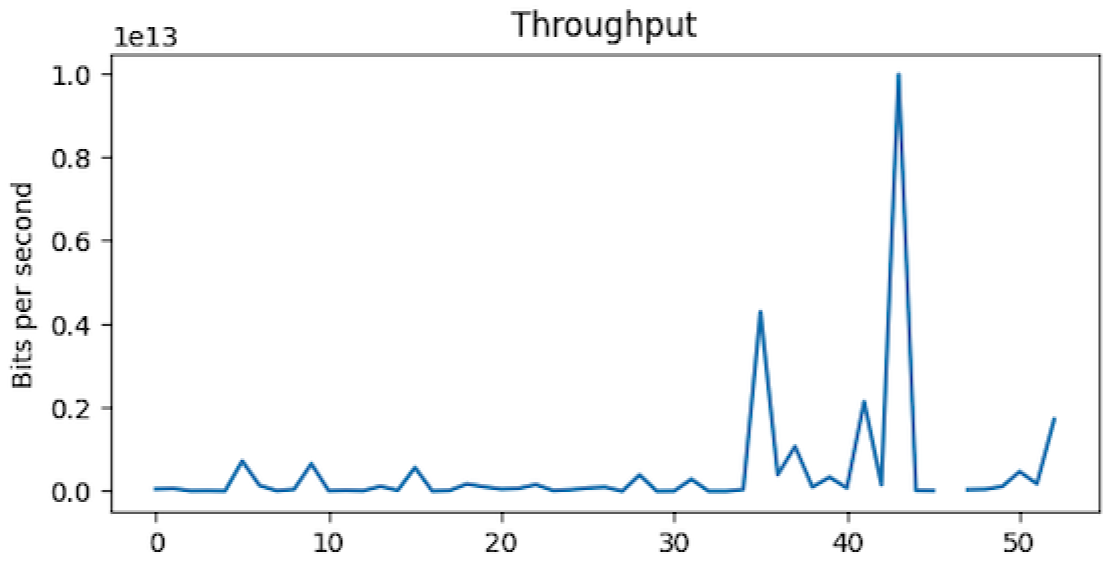
5. Performance metrics of Virgin media

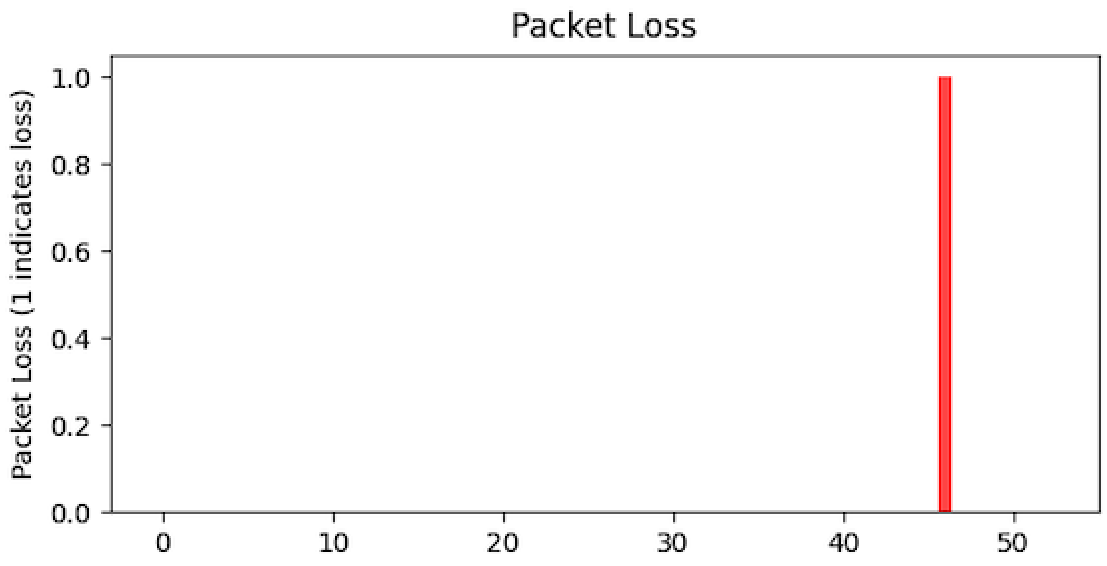


6. Performance metrics of KT




Jitter
1e6
Time (seconds)
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50


Throughput
1e13
Bits per second
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50


Packet Loss
Packet Loss (1 indicates loss)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50