

# **5G NETWORK SIMULATION USING NS3**

**A Major Project Report Submitted  
In partial fulfillment of the requirements for the award of the degree of**

## **Bachelor of Technology In Computer Science and Engineering (AIML)**

**By**

<b>K SAI BHUVANESHWARI</b>	<b>20N31A6629</b>
<b>M PAWAN KALYAN</b>	<b>20N31A6632</b>
<b>G BHARADWAJ</b>	<b>21N35A6601</b>

Under the esteemed guidance of

**Dr.G.L.N.JAYAPRADA**

**Assoc.Professor**



**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE  
MALLA REDDY COLLEGE OF ENGINEERING AND  
TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

**(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – ‘A’ Grade, ISO 9001:2015 Certified)  
Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India.**

**2023-2024**

# **5G NETWORK SIMULATION USING NS3**

**A Major Project Report Submitted  
In partial fulfillment of the requirements for the award of the degree of**

## **Bachelor of Technology In Computer Science and Engineering (AIML)**

**By**

**K SAI BHUVANESHWARI      20N31A6629**

**M PAWAN KALYAN      20N31A6632**

**G BHARADWAJ      21N35A6601**

Under the esteemed guidance of

**Dr.G.L.N JAYAPRADA**

**Assoc.Professor**



## **DEPARTMENT OF COMPUTATIONAL INTELLIGENCE MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

**(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – 'A' Grade, ISO 9001:2015 Certified)  
Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India.**

website: [www.mrcet.ac.in](http://www.mrcet.ac.in)



# **Malla Reddy College of Engineering and Technology**

**(Autonomous Institution-UGC, Govt. of India)**

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA&NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

Website: [www.mrcet.ac.in](http://www.mrcet.ac.in)

## **CERTIFICATE**

This is to certify that this is the Bonafide record of the project entitled “**5G NETWORK SIMULATION USING NS3**”, submitted by **K SAI BHUVANESHWARI (20N31A6629), M PAWAN KALYAN (20N31A6632)**

**G BHARADWAJ (21N35A6601)** of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering (AIML), Department of Computational Intelligence, during the year 2023-2024. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

**Project Guide**

**Dr. G.L.N JAYAPRADA**

**Assoc.Professor**

**Head of the Department**

**Dr. D. SUJATHA**

**Professor**

**External Examiner**

## **ACKNOWLEDGMENT**

I feel honored to place warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) for allowing me to do this Project as part of our B. Tech Program. We are ever grateful to our Director **Dr. V. S. K Reddy** and Principal **Dr. S. Srinivasa Rao** who enabled us to have experience in engineering and gain profound technical knowledge.

I express my heartiest thanks to our HOD, **Dr. D Sujatha** for encouraging me in every aspect of the course and helping me realize my full potential.

I would like to express my sincere gratitude and indebtedness to my project supervisor **Dr. G.L.N.Jayaprada** Assoc.Professor for her valuable suggestions and interest throughout the course of this project.

I would like to thank our Class in-charge, **Mr. D. Chandra Shekar Reddy** who despite being busy with her academic duties took time to guide and keep us on the correct path.

I convey my heartfelt thanks to our Project Coordinator **Dr. P. Harikrishna**, Assoc.Professor for his regular guidance and constant encouragement during our dissertation work.

I would also like to thank all the faculty members and supporting staff of the Department of CSE (AIML) and all other departments who have helped directly or indirectly in making our project a success.

I am extremely grateful to my parents for their blessings and prayers for the completion of the project, which gave us the strength to do my project.

With regards and gratitude

**K SAI BHUVANESHWARI                      20N31A6629**

**M PAWAN KALYAN                            20N31A6632**

**G BHARADWAJ                                21N35A6601**

## **ABSTRACT**

This project focuses on leveraging the capabilities of the ns-3 network simulator to analyze and simulate network behavior, particularly in the context of 5G networks. The project encompasses a comprehensive exploration of various aspects of network simulation, including network topology setup, application configuration, tracing mechanisms, and analysis of TCP congestion control behavior.

The project begins with an overview of the ns-3 simulation framework and its relevance in simulating 5G networks. It then delves into the setup of network topologies, covering scenarios involving point-to-point links, CSMA LANs, and wireless networks. Detailed explanations are provided on how to configure devices, set up IP addressing, and install applications for network traffic generation and analysis.

Furthermore, the project elucidates the importance of tracing mechanisms in network simulation and demonstrates how to define trace sources and connect them to trace sinks for capturing events and data changes. Various tracing methods, including Wireshark tracing, ASCII tracing, and PCAP tracing, are discussed, along with their applications in analyzing network behavior and performance.

Moreover, the project explores the behavior of TCP congestion control mechanisms in simulated networks, highlighting the significance of understanding and analyzing congestion window changes under varying network conditions. It presents a custom application, TutorialApp, for sending packets over a simulated network and implements trace sinks for monitoring congestion window changes and packet drops.

Overall, this project provides a comprehensive understanding of network simulation using ns-3, particularly in the context of 5G networks. It serves as a valuable resource for researchers, students, and network engineers looking to gain insights into network behavior, optimize network protocols, and evaluate network performance in simulated environments.

## TABLE OF CONTENTS

<b>S.NO</b>	<b>TITTLE</b>	<b>PG.NO</b>
<b>1</b>	<b>INTRODUCTION</b> 1.1 PURPOSE, AIM AND OBJECTIVE 1.2 EXISTING AND PROPOSED SYSTEM 1.3 SCOPE OF PROJECT	<b>01</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>16</b>
<b>3</b>	<b>TECHNOLOGIES USED</b> 3.1 NS3 3.2 NFV 3.3 SDN	<b>18</b>
<b>4</b>	<b>SYSTEM ANALYSIS</b> 4.1 HARDWARE AND SOFTWARE REQUIREMENTS 4.2 SOFTWARE REQUIREMENTS SPECIFICATION	<b>23</b>
<b>5</b>	<b>SYSTEM DESIGN</b> 5.1 DESCRIPTION 5.2 ARCHITECTURE 5.3 UML DIAGRAMS	<b>27</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>40</b>
<b>7</b>	<b>TESTING</b>	<b>44</b>
<b>8</b>	<b>OUTPUT SCREENS</b>	<b>54</b>
<b>9</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>71</b>
<b>10</b>	<b>BIBLIOGRAPHY</b>	<b>72</b>

# CHAPTER 1

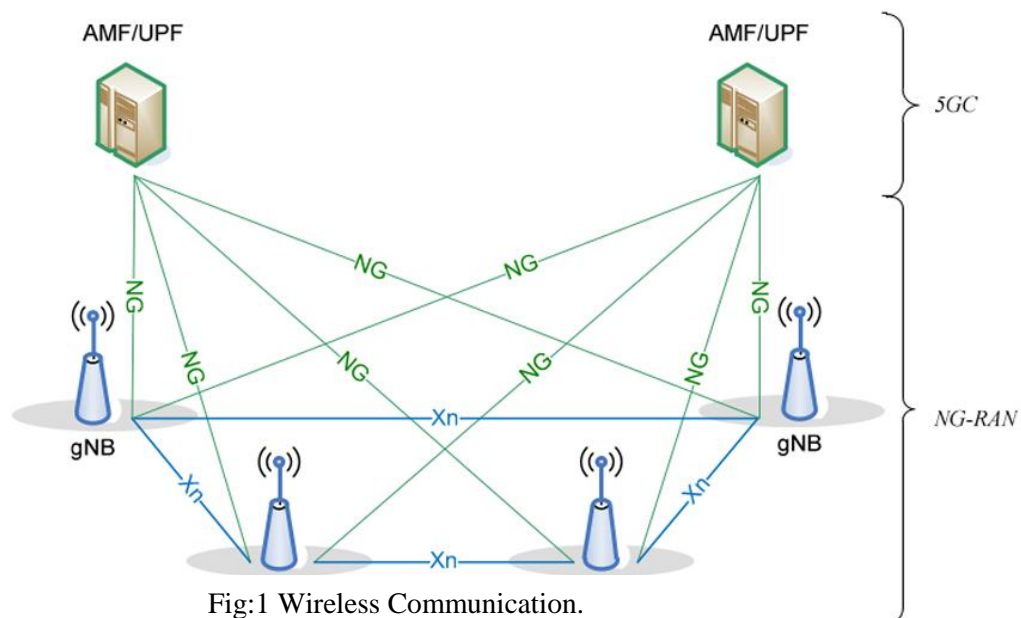
## INTRODUCTION

### 5G-Technology

#### Foundations of Wireless Communication

Wireless communication is the transmission of data over long distances without the need for physical cables. Understanding its foundational principles is essential for grasping the intricacies of 5G technology.

- **Transmission Mediums and Propagation:** Different transmission mediums, such as air or space, affect signal propagation. Understanding how signals propagate through various mediums is crucial.
- **Frequency Spectrum Allocation:** The allocation of frequency bands determines the capacity and performance of wireless communication systems.
- **Modulation Techniques:** Modulation schemes convert digital data into analog signals for transmission, with techniques like amplitude modulation (AM) and frequency modulation (FM).
- **Signal Encoding and Decoding:** Encoding and decoding methods .



## Radio Signal Properties

Radio signals are the carriers of wireless communication, transmitting data through electromagnetic waves. Understanding their properties is essential for optimizing signal transmission and reception.

- **Frequency and Wavelength:** Frequency determines the number of cycles per second, while wavelength is the distance between two consecutive peaks.
- **Amplitude and Phase Modulation:** Modulation techniques that alter the amplitude or phase of the carrier signal to encode data.
- **Signal Attenuation and Propagation Loss:** Signal attenuation refers to the weakening of signals over distance, influenced by factors like obstacles and interference.
- **Signal-to-Noise Ratio (SNR) and Signal Strength:** SNR measures the ratio of signal power to noise power, while signal strength indicates the power of the received signal.

## Factors Affecting Data Rate

The data rate, or throughput, of a wireless communication system depends on various factors. Understanding these factors is essential for optimizing network performance.

- **Channel Bandwidth and Spectral Efficiency:** Wider bandwidth allows for higher data rates, while spectral efficiency maximizes the data throughput within the available bandwidth.
- **Signal-to-Interference Ratio (SIR):** SIR measures the quality of the received signal relative to interference from other sources.
- **Multipath Propagation and Fading:** Multipath propagation occurs when signals reach the receiver via multiple paths, leading to fading and signal degradation.
- **Coding and Error Correction Techniques:** Error correction codes ensure data integrity by detecting and correcting errors that occur during transmission.

## General Concepts Behind Cellular Communication

Cellular communication divides geographical areas into cells served by base stations, enabling widespread wireless connectivity. Understanding its concepts is crucial for understanding the evolution to 5G networks.

- **Cell Layout and Frequency Reuse:** Cells are arranged in a grid pattern, with frequency reuse maximizing spectral efficiency.
- **Handover and Mobility Management:** Handover ensures seamless transition between cells.



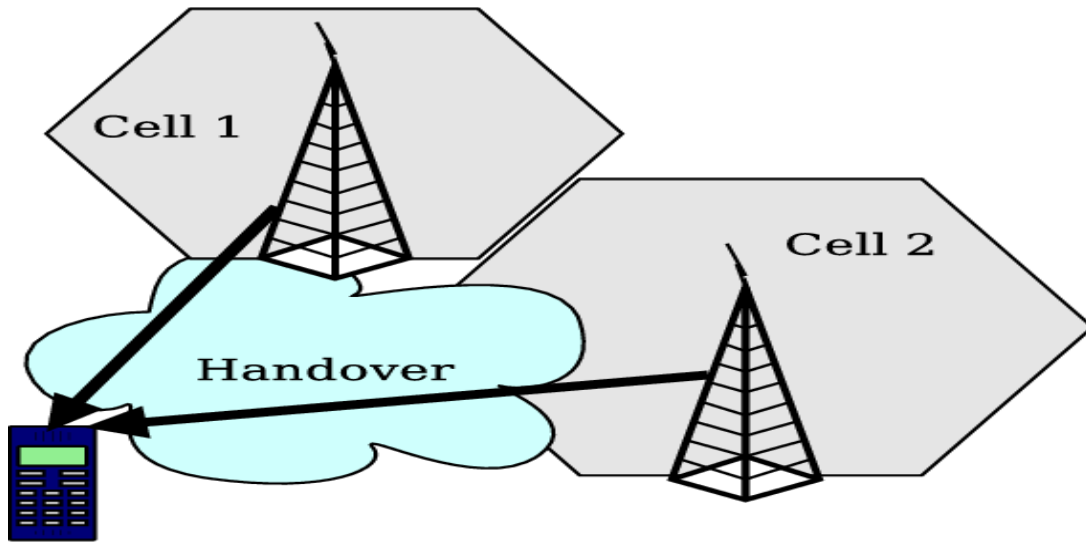


Fig 2: Cellular Communication.

- **Cell Site Infrastructure and Network Topology:** Cell sites consist of base stations, antennas, and other infrastructure components, interconnected to form a cellular network.
- **Cellular Protocols and Standards:** Various protocols and standards govern cellular communication, ensuring interoperability and compatibility between different network elements.

### 5G NR Vision and Capabilities

5G New Radio (NR) represents the next evolution in wireless technology, promising transformative capabilities to meet growing demands for connectivity.

- **Ultra-High Data Rates:** 5G offers significantly higher data rates compared to previous generations, enabling bandwidth-intensive applications like ultra-HD video streaming and virtual reality.
- **Ultra-Low Latency:** 5G aims to minimize latency, reducing the time it takes for data to travel between devices and networks.
- **Massive Device Connectivity:** 5G supports a massive number of connected devices, catering to the growing Internet of Things (IoT) ecosystem.
- **Enhanced Reliability:** 5G networks prioritize reliability, ensuring consistent connectivity for mission-critical applications like autonomous vehicles and industrial automation.

### Why We Need 5G

The proliferation of smart devices, IoT applications, and emerging technologies necessitates the deployment of 5G networks to address evolving connectivity demands.

- **Growing Bandwidth Demand:** Increasing usage of bandwidth-intensive applications requires higher data rates and network capacity.
- **Emerging Use Cases:** New applications like autonomous vehicles, remote healthcare, and smart cities require ultra-low latency and reliable connectivity.
- **IoT Expansion:** The IoT ecosystem continues to expand, with billions of connected devices requiring efficient and scalable connectivity solutions.
- **Industry Advancements:** Industries such as manufacturing, transportation, and energy are leveraging connectivity for automation and optimization.

### Fundamental Technological Capabilities of 5G

Key technological advancements in 5G enable its transformative capabilities, ranging from massive MIMO to edge computing.

- **Massive MIMO:** Utilizes multiple antennas to improve spectral efficiency and throughput.
- **Millimeter Wave Spectrum:** Exploits higher frequency bands for increased capacity and data rates.
- **Network Slicing:** Enables the creation of virtualized network slices tailored to specific services or applications.
- **Software-Defined Networking (SDN) and Network Function Virtualization (NFV):** Enhances network flexibility, scalability, and efficiency.
- **Edge Computing:** Moves computing resources closer to the user for reduced latency and improved performance.

### Key Service Classes of 5G

5G caters to diverse use cases through three main service classes: Enhanced Mobile Broadband (eMBB), Massive Machine Type Communications (mMTC), and Ultra-Reliable Low-Latency Communications (URLLC).

- **Enhanced Mobile Broadband (eMBB):** Provides high data rates and capacity for applications like ultra-HD video streaming and immersive gaming.
- **Massive Machine Type Communications (mMTC):** Supports a massive number of IoT devices with low-power and low-complexity connectivity.

- **Ultra-Reliable Low-Latency Communications (URLLC):** Ensures ultra-low latency and high reliability for mission-critical applications such as autonomous vehicles and industrial automation.

## **1.1 PURPOSE AND OBJECTIVE**

The purpose of this project is to develop a comprehensive network simulation system using the ns-3 framework. The aim is to provide users with a tool for simulating and analyzing various network scenarios to understand protocol behavior, packet flows, and network performance. The objectives include enabling Wireshark tracing, implementing ASCII tracing, visualizing simulations with NetAnim, and understanding TCP congestion control behavior.

### **5G Layers and Architecture:**

The architecture of 5G networks is designed to meet the diverse requirements of emerging use cases, ranging from enhanced mobile broadband to ultra-reliable low-latency communications. This documentation provides a detailed overview of the layers and architecture of 5G technology, encompassing both the radio access network (RAN) and the core network.

#### **1. 5G Protocol Stack Layers**

The 5G protocol stack comprises multiple layers, each serving a specific function in the transmission and reception of data. Understanding these layers is essential for comprehending the flow of information within the 5G network.

#### **Layers:**

1. **Physical Layer (PHY):** Responsible for transmitting and receiving the physical signals over the air interface. It includes modulation, coding, and transmission techniques.
2. **Medium Access Control (MAC) Layer:** Controls access to the shared radio resources and manages data transmission between the physical layer and higher layers.
3. **Radio Link Control (RLC) Layer:** Provides reliable data transfer services, segmentation, reassembly, and error correction functionalities.
4. **Packet Data Convergence Protocol (PDCP) Layer:** Handles the compression and decompression of user data, header compression, and encryption.
5. **Radio Resource Control (RRC) Layer:** Manages the radio resources, mobility, and connection establishment between the user equipment (UE) and the base station.

## 2. 5G Network Architecture

The architecture of 5G networks is based on a flexible and scalable design, facilitating the delivery of diverse services and applications with varying requirements. It consists of the radio access network (RAN) and the core network (CN), interconnected to provide seamless connectivity.

### **Radio Access Network (RAN):**

The RAN is responsible for connecting user equipment (UE) to the core network and includes the following components:

1. **g NB (Next-Generation Node B):** The g NB is the base station in 5G networks, responsible for transmitting and receiving radio signals to and from UEs.
2. **Centralized Unit (CU):** The CU handles centralized processing functions such as scheduling, coordination, and management of radio resources.
3. **Distributed Unit (DU):** The DU performs distributed processing functions and interfaces with the g NB to process radio signals.

### **Core Network (CN):**

The core network provides various network functions and services to support communication between UEs and external networks. It consists of the following elements:

1. **User Plane Function (UPF):** The UPF handles user data forwarding and traffic management functions in the data plane.
2. **Control Plane Function (CPF):** The CPF manages control signaling and session management functions in the control plane.
3. **Session Management Function (SMF):** The SMF manages session establishment, mobility management, and quality of service (QoS) enforcement.
4. **Access and Mobility Management Function (AMF):** The AMF handles access authentication, mobility management, and security functions.
5. **Authentication Server Function (AUSF):** The AUSF provides authentication and key management services for secure access to the network.

The layered architecture of 5G networks, encompassing both the radio access network and the core network, is designed to deliver high performance, flexibility, and scalability to meet the diverse requirements of emerging use cases. Understanding the layers and architecture is essential for network designers, engineers, and operators to deploy and manage 5G networks effectively.

## 1.2 EXISTING AND PROPOSED SYSTEM:

### Existing System: G Node B to UE Packet Flow in 5G Network

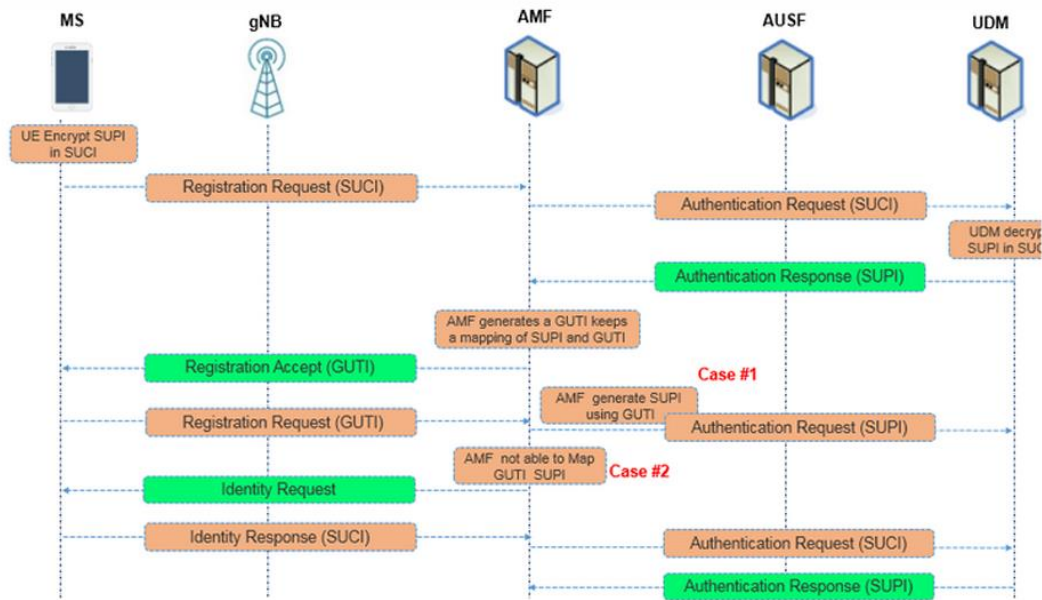


Fig 3: Packet Flow.

In a 5G network architecture, the g Node B (gNB) serves as the base station responsible for communicating with User Equipment (UE) and facilitating data transmission. The packet flow from the gNB to the UE involves several steps, encompassing various protocols and procedures to ensure efficient and reliable communication.

In the 5G ecosystem, the exchange of identities between the User Equipment (UE) and the network plays a critical role in establishing secure and reliable communication. This process involves several network entities and procedures to ensure the confidentiality and integrity of user information. Let's delve into a detailed documentation of the identity exchange process:

#### 1. UE (User Equipment):

- The UE represents the mobile device or equipment used by the end-user to connect to the 5G network and access telecommunications services.

#### 2. gNB (Next-Generation NodeB):

- The gNB is a key component of the 5G radio access network responsible for managing radio resources and facilitating communication between the UE and the core network.

**3. AMF (Access and Mobility Management Function):**

- The AMF is a core network entity responsible for access and mobility management functions, including authentication and session management for UE connections.

**4. AUSF (Authentication Server Function):**

- The AUSF is responsible for generating authentication vectors and verifying the authenticity of the UE during the authentication process.

**5. UDM (Unified Data Management):**

- The UDM stores and manages subscriber-related data, including authentication credentials and user profiles, essential for authentication and authorization processes.

**Identity Exchange Process:**

**1. UE Encrypts SUPI in SUCI:**

- The UE encrypts its Subscription Permanent Identifier (SUPI) using a Temporary Identifier called Subscription Concealed Identifier (SUCI) to protect its identity during communication with the network.

**2. Registration Request (SUCI):**

- The UE sends a registration request to the network, including the encrypted SUCI, to initiate the registration process.

**3. Authentication Request (SUCI):**

- Upon receiving the registration request, the AMF forwards an authentication request containing the SUCI to the AUSF for authentication and verification.

**4. Authentication Response (SUPI):**

- The AUSF validates the SUCI and returns an authentication response containing the decrypted SUPI to the AMF.

**5. UDM Decrypts SUPI in SUK:**

- The AMF forwards the SUPI to the UDM, where it is decrypted using the Subscription Key (SUK) stored in the subscriber's profile.

**6. AMF Generates a GUTI and Maps SUPI and GUTI:**

- Upon successful authentication, the AMF generates a Globally Unique Temporary Identifier (GUTI) for the UE and maintains a mapping between the SUPI and GUTI for future reference.

**7. Registration Accept (GUTI):**

- The AMF sends a registration accept message to the UE, including the assigned GUTI, confirming the successful registration process.

## **8. Registration Request (GUTI):**

- In subsequent registration requests, the UE uses the assigned GUTI instead of the SUCI for identity exchange with the network.

### **Identity Resolution Scenarios:**

#### **1. Case #1 - AMF Generates SUPI using GUTI:**

- In scenarios where the AMF needs to generate the SUPI using the GUTI, it initiates an authentication request containing the GUTI to the AUSF.

#### **2. Identity Request:**

- The AUSF requests the UE's identity from the AMF when it receives a GUTI instead of the SUPI during authentication.

#### **3. Identity Response (SUCI):**

- If the AMF cannot map the GUTI to a valid SUPI, it responds with an Identity Response containing a SUCI, prompting the UE to provide its SUPI.

#### **4. Case #2 - Authentication Request (SUCI):**

- In cases where the AMF successfully maps the GUTI to a SUPI, it proceeds with the authentication request using the SUPI.

#### **5. Authentication Response (SUPI):**

- The AUSF validates the SUPI and responds with an authentication response containing the decrypted SUPI to the AMF for further processing.

Through this detailed documentation, we gain insights into the intricate identity exchange process between the UE and the 5G network, highlighting the crucial role of various network entities and procedures in ensuring secure communication and user privacy.

### **Packet Data Convergence Protocol (PDCP) Layer Architecture:**

The Packet Data Convergence Protocol (PDCP) layer operates within the 3GPP network architecture, serving as a crucial element in the transmission and reception of data packets. Its architecture can be understood from both structural and functional perspectives.

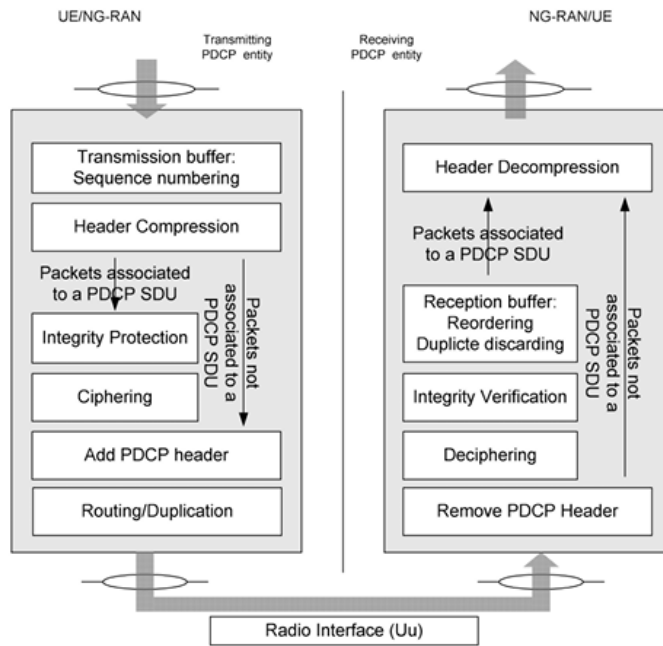


Fig 4: PDCP Layer Architecture

## PDCP Layer Structural View

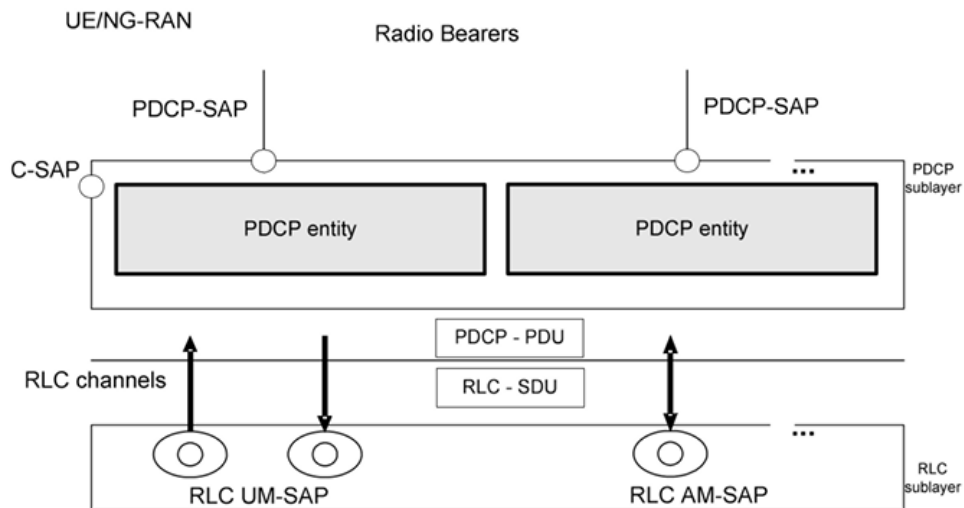


Fig 5: PDCP Layer Structural View

In the structural view, the PDCP layer comprises various components organized in a hierarchical manner to facilitate efficient data transfer and management. Figure [1] illustrates the structural layout of the PDCP layer, depicting the relationships and interactions between its constituent parts.



## **PDCP Layer Functional View**

From a functional standpoint, the PDCP layer is responsible for providing essential services to higher layers, such as the Radio Resource Control (RRC) or Service Data Adaptation Protocol (SDAP) layers. These services include header compression, ciphering, integrity protection, and the transfer of both user plane and control plane data.

## **Services Provided by PDCP Layer**

The PDCP layer offers a range of services to higher layers within the protocol stack. These services encompass header compression to reduce overhead, ciphering for data security, integrity protection to ensure data authenticity, and the reliable transfer of both user plane and control plane data. Notably, the PDCP layer supports a maximum SDU size of 9000 bytes and a maximum Control PDU size of 9000 bytes, accommodating diverse data requirements.

## **Functions of PDCP Layer**

Among its functions, the PDCP layer undertakes header compression and decompression using the Robust Header Compression (ROHC) protocol, ciphering and deciphering to safeguard data integrity, integrity protection and verification mechanisms, routing for split bearers, and duplication control to manage redundant data streams efficiently.

## **Procedures within PDCP Layer**

Key procedures within the PDCP layer include entity handling, data transfer management, data recovery in case of transmission errors, header compression, decompression, ciphering, deciphering, integrity protection, and verification.

## **PDCP Entity Handling**

Entity handling within the PDCP layer involves the establishment, re-establishment, release, and suspension of PDCP entities, each operation delineated between transmitting and receiving entities to ensure seamless communication.

## **Data Transfer Operations**

Data transfer operations within the PDCP layer encompass both transmit and receive operations, facilitating the exchange of data between transmitting and receiving entities while adhering to defined protocols and procedures.

## **Transmit Operation**

The transmit operation encompasses various sub-operations for both upper and lower layers of the PDCP. For upper layers, these sub-operations include sequence numbering, header compression, integrity protection, ciphering, and submission of PDCP data to lower layers.

## **Sequence Numbering**

Sequence numbering involves the association of a COUNT value with each PDCP SDU to be transmitted, facilitating efficient data sequencing and management. Additionally, a discardTimer is initiated for data retention purposes, particularly relevant for Data Radio Bearers (DRBs).

## **Receive Operation**

The receive operation involves actions upon receiving PDCP Data PDUs from lower layers, including handling reordering, verifying integrity protection, and managing associated parameters such as HFN, SN, and RCVD-SN.

## **Parameters and State Variables**

Various parameters and state variables are utilized during PDCP operations, including HFN, SN, RCVD-SN, RCVD-HFN, RCVD-COUNT, and others, to ensure accurate data processing and transmission.

## **Supported Header Compression Protocols**

The PDCP layer supports multiple header compression protocols and profiles based on the ROHC framework, tailored to specific network, transport, or upper layer protocol combinations, enhancing efficiency and reducing overhead.

## **Protocol Data Units (PDUs) and Formats**

PDUs within the PDCP layer are structured into data and control units, each adhering to defined formats and specifications to facilitate standardized communication and interoperability.

## **Constants and Timers**

The PDCP layer employs constants such as Window-Size and timers including discardTimer and t-Reordering to regulate data flow, manage reordering, and ensure timely processing of packets.

The architecture of the PDCP layer is intricately designed to facilitate efficient data transmission, compression, security, and integrity verification within the 3GPP network. Understanding its structural layout, functional components, and operational procedures is essential for optimizing network .

## Proposed System:

### Overview of Security Architecture

The security architecture illustrated in the provided diagram serves as a comprehensive framework for ensuring the integrity, confidentiality, and secure transmission of data within a telecommunications network. Let's delve into a detailed documentation of each component and layer depicted in the diagram:

1. **User Application (IV):** At the forefront of the architecture lies the User Application layer, denoted by "IV". This layer encompasses a myriad of user-facing applications and services, ranging from messaging apps to multimedia streaming platforms, catering to the diverse needs of end-users.
2. **Provider Application:** Mirroring the User Application layer is the Provider Application layer, which represents the suite of applications and services offered by the telecommunications service provider. These applications could include billing systems, customer support portals, and network management tools, aimed at facilitating seamless service delivery and network operation.
3. **Application Stratum:** The Application Stratum encompasses both the User and Provider applications, symbolizing the critical interface where user-generated data interacts with the provider's infrastructure. Securing this stratum is imperative to protect sensitive user information and ensure the reliability of telecommunications services.
4. **Mobile Equipment (ME):** Positioned within the user's domain is the Mobile Equipment (ME), representing the mobile device or equipment utilized by the end-user to access telecommunications services. This could include smartphones, tablets, or IoT devices equipped with wireless connectivity capabilities.
5. **Universal Subscriber Identity Module (USIM):** Integral to the mobile device is the Universal Subscriber Identity Module (USIM), a smart card housing subscriber information and cryptographic keys essential for authentication, encryption, and secure communication between the device and the network.
6. **Home Stratum/Serving Stratum:** Transitioning into the network infrastructure, we encounter the Home Stratum or Serving Stratum, comprising core network elements responsible for managing user sessions, authentication, and authorization. These elements ensure seamless connectivity and service delivery while upholding stringent security measures.
7. **Home Environment (HE):** At the core of the Home Stratum lies the Home Environment (HE), a pivotal component responsible for authenticating mobile subscribers, managing subscriber profiles, and coordinating mobility management functions within the network.

8. **Serving Network (SN):** Intersecting with the Home Stratum is the Serving Network (SN), which represents the segment of the core network tasked with serving mobile subscribers within a specific geographical area. This network segment ensures efficient routing and delivery of data packets to and from mobile devices.
9. **3GPP Access Network (3GPP AN):** Extending beyond the core network, the 3rd Generation Partnership Project (3GPP) Access Network comprises the infrastructure elements deployed by service providers, such as LTE or 5G networks, to provide wireless connectivity to mobile devices.
10. **Non-3GPP Access Network:** In addition to 3GPP standards, the architecture acknowledges the presence of Non-3GPP Access Networks, encompassing alternative wireless technologies utilized for connecting mobile devices, such as Wi-Fi or satellite networks.
11. **Transport Stratum:** Serving as the underlying foundation for data transmission, the Transport Stratum encapsulates the transport layer protocols and technologies responsible for securely ferrying data packets between network elements. This layer ensures reliable and efficient communication while mitigating security risks.

### Optimising Packet Flow in 5G Networks using NS3

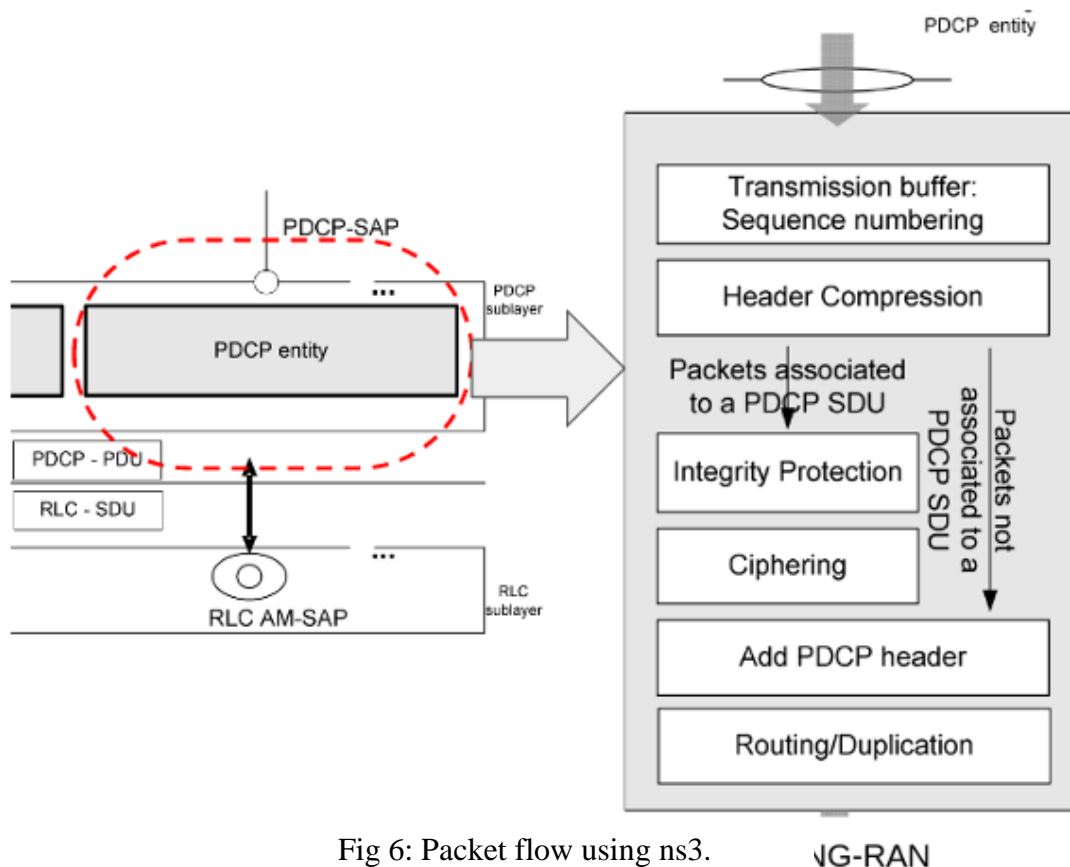


Fig 6: Packet flow using ns3.

In 5G networks, the transmission of data packets from gNodeB to User Equipment (UE) involves traversing various protocol layers, each responsible for specific functions to ensure reliable and secure communication. However, in certain scenarios, such as optimizing packet flow for efficiency or reducing processing overhead, it may be beneficial to bypass certain layers in the communication stack. In this documentation, we explore the process of streamlining packet flow from gNodeB to UE by skipping the Ciphering, Deciphering, Integrity Protection, and Integrity Verification layers, and establishing a direct connection between Header Compression and PDCP layers.

The 5G protocol stack comprises several layers, including the Physical (PHY), Medium Access Control (MAC), Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP), Radio Resource Control (RRC), and application layers. Each layer plays a crucial role in data transmission, ensuring reliability, security, and efficient resource utilization.

In our scenario, we aim to streamline the packet flow by removing the Ciphering, Deciphering, Integrity Protection, and Integrity Verification layers. These layers are primarily responsible for securing the communication channel through encryption and ensuring data integrity. By bypassing these layers, we can reduce processing overhead and potentially improve throughput.

To simulate and analyze the optimized packet flow in 5G networks, we will leverage the Network Simulator 3 (NS3). NS3 is a discrete-event network simulator capable of modeling complex network scenarios and protocols. It provides a comprehensive platform for researchers and developers to evaluate network performance, conduct experiments, and develop network applications.

Using NS3, we can design a network topology that represents the 5G infrastructure, including gNodeB, UE, and the communication channels between them. By customizing the protocol stack in NS3, we can selectively disable the Ciphering, Deciphering, Integrity Protection, and Integrity Verification layers, ensuring that data packets flow directly from Header Compression to the PDCP layer.

By streamlining packet flow and removing unnecessary protocol layers, several benefits can be realized. These include reduced latency, improved resource utilization, and enhanced network performance. Additionally, simplified packet processing can lead to lower energy consumption, making the network more energy-efficient.

In conclusion, the process of optimizing packet flow in 5G networks by bypassing certain protocol layers and establishing a direct connection between Header Compression and the PDCP layer presents a promising avenue for enhancing network efficiency. Leveraging the capabilities of NS3, researchers and

developers can conduct in-depth analysis, evaluate performance metrics, and innovate novel solutions to propel the evolution of 5G communication systems towards unprecedented levels of efficiency and reliability.

In the context of analyzing and simulating telecommunications networks like 5G, the Network Simulator 3 (NS3) emerges as a powerful tool. NS3 is an open-source discrete-event network simulator used for research and development in various network domains, including wireless, wired, and Internet protocols. It provides a flexible and extensible platform for modeling complex network scenarios, evaluating network performance, and testing new protocols and algorithms. Researchers and developers leverage NS3 to simulate real-world network environments, analyze protocol behaviors, and optimize network.

### **1.3 SCOPE OF THE PROJECT:**

The scope of this project encompasses an in-depth exploration of network simulation using the ns-3 framework with a specific focus on 5G network technologies. The project aims to provide a detailed understanding of network simulation methodologies, tools, and techniques, along with their practical applications in the context of 5G networks.

#### **1. Simulation Environment Setup:**

- The project will cover the setup and configuration of the ns-3 simulation environment, including the installation of necessary modules and libraries.
- Special emphasis will be placed on configuring ns-3 for simulating 5G network scenarios, such as mmWave communication, massive MIMO, and ultra-reliable low-latency communication (URLLC).

#### **2. Network Topology Design:**

- Various network topologies relevant to 5G networks will be designed and simulated using ns-3.
- This includes scenarios involving heterogeneous networks (HetNets), dynamic TDD/FDD configurations, and network slicing.

#### **3. Protocol Implementation and Evaluation:**

- Implementation and evaluation of 5G protocol stacks, including NR (New Radio) and NGAP (Next Generation Application Protocol).
- Performance evaluation of key 5G features such as beamforming, carrier aggregation, and QoS provisioning.

#### **4. Mobility Modeling and Simulation:**

- Integration of realistic mobility models for simulating mobile users and devices in 5G networks.

- Study of mobility management protocols and handover procedures in a 5G context.

#### **5. Traffic Modeling and Analysis:**

- Generation of realistic traffic patterns and application scenarios to simulate diverse use cases in 5G networks.
- Analysis of traffic characteristics, such as throughput, latency, and packet loss, under varying network conditions.

#### **6. Integration with Real-World Deployments:**

- Exploration of techniques for integrating ns-3 simulations with real-world 5G deployments and testbeds.
- Validation of simulation results through comparison with empirical data collected from operational 5G networks.

#### **7. Performance Optimization and Scalability:**

- Investigation of methods for optimizing simulation performance and scalability to handle large-scale 5G network scenarios efficiently.
- Analysis of resource utilization and computational overhead in ns-3 simulations of 5G networks.

#### **8. Visualization and Reporting:**

- Development of visualization tools and techniques for presenting simulation results in a comprehensive and intuitive manner.
- Preparation of detailed reports and documentation summarizing the findings, insights, and implications of the simulated 5G network scenarios.

## **CHAPTER 2**

### **LITERATURE SURVEY**

**[1] Andrews, Jeffrey G. - "Faster Data Speeds: Advancements in 5G Technology"**

Jeffrey G. Andrews' research focuses on the advancements in data speeds brought about by 5G technology. With the potential to achieve peak speeds of up to 20 Gbps, 5G networks offer a significant improvement over previous generations. This capability enables users to enjoy high-definition video streaming, real-time gaming, and seamless multimedia experiences on their devices.

**[2] Ghosh, Amitava - "Ultra-Low Latency: Enhancing Connectivity with 5G"**

Amitava Ghosh's research highlights the ultra-low latency capabilities of 5G networks and their impact on connectivity. With latency as low as 1 millisecond, 5G technology opens up possibilities for applications such as autonomous vehicles, remote surgery, and augmented reality. This ultra-low latency ensures minimal delay in data transmission, enhancing user experiences and enabling real-time interactions.

**[3] Rappaport, Theodore S. - "Massive IoT Connectivity: Scaling with 5G Networks"**

Theodore S. Rappaport's research delves into the scalability of 5G networks in supporting massive IoT connectivity. With the ability to accommodate up to 1 million devices per square kilometer, 5G networks empower various sectors such as smart cities, industrial automation, and smart agriculture. This scalability enables seamless communication and coordination among a multitude of IoT devices, driving efficiency and innovation.

**[4] Zhang, Jianzhong - "Addressing Infrastructure Challenges in 5G Deployment"**

Jianzhong Zhang's research focuses on the infrastructure challenges associated with deploying 5G networks. This includes the need to densify network coverage, optimize spectrum allocation, and ensure seamless connectivity across diverse geographical areas. Addressing these challenges is essential for the successful deployment and operation of 5G networks, enabling reliable and high-performance wireless.



#### **[5] Boccardi, Federico - "Spectrum Allocation and Interference Management in 5G"**

Federico Boccardi's research explores the complexities of spectrum allocation and interference management in 5G networks. Effectively managing spectrum resources and mitigating interference are crucial for optimizing network performance, particularly in densely populated urban environments. This research investigates techniques and strategies to ensure efficient spectrum utilization and minimize interference, enhancing the quality of service for users.

#### **[6] Dhillon, Harpreet Singh - "Security and Privacy Concerns in 5G Networks"**

Harpreet Singh Dhillon's research highlights the security and privacy concerns associated with 5G networks. As 5G facilitates the exchange of sensitive data across a multitude of devices, robust security protocols and privacy protections are essential to prevent security breaches, data breaches, and unauthorized access. This research explores vulnerabilities inherent in 5G infrastructure and proposes strategies to enhance security and privacy safeguards.

#### **[7] Wang, Yingying - "Future Directions: Network Slicing and Virtualization"**

Yingying Wang's research discusses future directions for 5G technology, with a focus on network slicing and virtualization. Network slicing enables the creation of virtualized network instances tailored to specific applications or user requirements, allowing for dynamic resource allocation and service customization. This research explores the potential of network slicing to enhance quality of service and support diverse use cases in 5G networks.

#### **[8] Sun, Qianmu - "Global Collaboration and Regulatory Frameworks for 5G"**

Qianmu Sun's research discusses the importance of global collaboration and regulatory frameworks for the successful deployment and operation of 5G networks worldwide. Standardization bodies and regulatory authorities play a crucial role in establishing interoperability standards, spectrum policies, and security protocols. This research emphasizes the need for international cooperation and regulatory harmonization to ensure the seamless adoption and advancement of 5G technology on a global scale.

## **CHAPTER 3**

### **TECHNOLOGIES USED**

## **Ns-3:**

### **Introduction**

The NS-3 simulator stands as a versatile discrete-event network simulator, primarily intended for research and educational purposes. Beginning its journey in 2006, the NS-3 project has evolved into a robust open-source platform tailored for network simulations.

### **About NS-3**

NS-3 emerges as a pivotal tool in network simulation, providing an open, extensible platform for both research and educational endeavours. Essentially, NS-3 furnishes models depicting the dynamics of packet data networks, coupled with a robust simulation engine for conducting experiments. Its utility spans various domains, enabling studies that may be impractical or infeasible in real-world scenarios, facilitating controlled and reproducible environment studies, and offering invaluable insights into network behaviour.

While NS-3 primarily focuses on Internet protocols and networks, its versatility extends to modelling non-Internet-based systems, catering to diverse research interests.

### **Key Features**

- **Modular Architecture:** NS-3 is designed as a collection of libraries that can be seamlessly integrated with external software libraries, fostering a modular and extensible environment.
- **Platform Compatibility:** While primarily supported on Linux and macOS systems, NS-3 offers compatibility with BSD systems and certain Windows frameworks. However, native support for Windows Visual Studio is currently unavailable.
- **Community Support:** As an open-source project, NS-3 thrives on community contributions and support. Users can seek assistance and engage with the community through the NS-3-users forum.

### **NS-3 Project Installation Guide**

The NS-3 simulator relies on several third-party libraries, but most of NS-3 can be built and utilised with support for common components. The following prerequisites are required to get started with NS-3:

C++ Compiler: Either clang or g++ (version 9 or greater)  
Python: python3 version >=3.6  
CMake: cmake version >=3.13  
Build System: make, ninja, xcodebuild (XCode)  
Git: any recent version (to access NS-3 from GitLab.com)  
tar: any recent version (to unpack an NS-3 release)  
bunzip2: any recent version (to uncompress an NS-3 release)

## Step 1: Package Installation

To ensure smooth operation and avoid potential bugs, it's recommended to install the following packages. While not all of them are compulsory, having them installed will enhance your NS-3 experience.

- `sudo apt update`
- `sudo apt install g++ python3 cmake ninja-build git gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 tcpdump wireshark sqlite sqlite3 libsqlite3-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools openmpi-bin openmpi-common openmpi-doc libopenmpi-dev doxygen graphviz imagemagick python3-sphinx dia imagemagick texlive dvipng latexmk texlive-extra-utils texlive-latex-extra texlive-font-utils libeigen3-dev gsl-bin libgsl-dev libgslcblas0 libxml2 libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-utilities ebttables bridge-utils libboost-all-dev`

## Step 2: Extract NS-3 Source

After ensuring all necessary packages are installed, proceed to extract the NS-3 source archive to your desired location. In this example, we'll assume it's the home folder.

- `cd /home/pradeepkumar`
- `tar xjf ns-allinone-3.38.tar.bz2`

## Step 3: Build NS-3

Once the source code is extracted, navigate to the NS-3 directory and initiate the build process.

- `cd ns-allinone-3.38/`
- `./build.py --enable-examples --enable-tests`

## Download the NR Module:

- `cd contrib`
- `git clone https://gitlab.com/cttc-lena/nr.git`
- `cd nr`

## Checkout the Compatible Version of NR:

`git checkout -b 5g-lena-v3.0.y origin/5g-lena-v3.0.y`

### **Testing NS-3 + NR Installation**

Once the NR module is downloaded, proceed with testing the NS-3 + NR installation:

#### **Configure the NS-3 + NR Project:**

```
./ns3 configure --enable-examples --enable-tests
```

Ensure that the output displays the following:

SQLite support: ON

Eigen3 support: ON

If these options are not enabled, revisit the "NS-3 and NR Prerequisites" section and install any missing prerequisites. After installing the required packages, rerun the configuration command.

#### **Compile NS-3 with NR:**

- `./ns3 build`

Upon successful recognition of the NR module, you should see "nr" listed in the built modules.

### **NFV:**

Network Function Virtualization (NFV) is a paradigm shift in the way network services are deployed, managed, and operated. It involves decoupling network functions from dedicated hardware appliances and instead running them as software instances on standard hardware infrastructure. This transformation enables greater flexibility, scalability, and efficiency in delivering network services.

NFV leverages virtualization technologies to abstract network functions from the underlying hardware, essentially turning them into software-defined entities. Traditionally, network functions such as firewalls, routers, load balancers, and intrusion detection systems were implemented using specialized hardware appliances, each serving a specific purpose. However, this approach led to inflexible and costly network deployments, as each function required its dedicated hardware.

With NFV, these network functions are virtualized and consolidated onto standard server hardware, often deployed in data centers or cloud environments. This consolidation eliminates the need for dedicated hardware appliances, reducing hardware costs, power consumption, and physical footprint. Moreover, NFV enables dynamic allocation of resources, allowing network functions to scale up or down based on demand.

#### **Key components of NFV include:**

**Virtualized Network Functions (VNFs):** These are the software implementations of network functions that run on virtualized infrastructure. VNFs perform the same tasks as their hardware counterparts but offer greater flexibility and scalability.

NFV Infrastructure (NFVI): This refers to the underlying hardware and software infrastructure that hosts VNFs. NFVI typically comprises standard server hardware, virtualization platforms (e.g., hypervisors), and software-defined networking (SDN) controllers.

NFV Management and Orchestration (NFV MANO): NFV MANO is responsible for the lifecycle management of VNFs, including deployment, scaling, monitoring, and optimization. It comprises three main components: Virtualized Infrastructure Manager (VIM), NFV Orchestrator (NFVO), and Virtualized Network Function Manager (VNFM).

### **Benefits of NFV:**

**Flexibility:** NFV enables service providers to deploy and manage network functions in a more agile and flexible manner. VNFs can be dynamically instantiated, migrated, or decommissioned in response to changing traffic patterns and service demands.

**Scalability:** NFV allows for elastic scaling of network functions, enabling service providers to allocate resources based on demand. This scalability ensures efficient resource utilization and optimal performance under varying workload conditions.

**Cost Reduction:** By virtualizing network functions and consolidating them onto standard hardware infrastructure, NFV reduces hardware costs, power consumption, and operational expenses. Service providers can achieve significant cost savings while maintaining or improving service quality.

**Service Innovation:** NFV accelerates service innovation by enabling rapid deployment of new network services and functionalities. Service providers can introduce new services more quickly and efficiently, staying ahead of market trends and customer demands.

### **SDN:**

Software-Defined Networking (SDN) is a network architecture approach that separates the control plane from the data plane, enabling centralized management and programmability of network resources. Unlike traditional networking architectures, where network devices make forwarding decisions locally, SDN centralizes control in a software-based controller, which communicates with network devices through open interfaces.

Here's a detailed explanation of SDN:

#### **Separation of Control Plane and Data Plane:**

- In SDN, the control plane, responsible for making decisions about how traffic should be forwarded, is decoupled from the data plane, which is responsible for forwarding packets based on those decisions. This separation allows for centralized control and programmability of

network behavior.

**SDN Controller:**

- At the heart of an SDN architecture is the SDN controller, which serves as the centralized brain of the network. The controller communicates with network devices, such as switches and routers, using open protocols like OpenFlow, to instruct them on how to handle traffic flows. It maintains a global view of the network topology and traffic patterns, making intelligent decisions about traffic forwarding and network management.

**Programmability:**

- SDN enables network administrators to program and customize network behavior through software interfaces exposed by the SDN controller. This programmability allows for dynamic configuration of network policies, traffic engineering, and service chaining, based on application requirements and network conditions.

**Centralized Management and Orchestration:**

- By centralizing control, SDN simplifies network management and orchestration tasks. Network policies and configurations can be defined and enforced globally from the SDN controller, reducing the complexity of managing individual network devices.

**Dynamic Network Slicing:**

- SDN enables dynamic network slicing, where virtualized network instances, or slices, are created to meet the specific requirements of different applications or tenants. Each network slice can have its own isolated set of resources, policies, and performance characteristics, enabling efficient resource sharing and multi-tenancy.

**Traffic Engineering and Optimization:**

- SDN facilitates dynamic traffic engineering and optimization by intelligently routing traffic based on real-time network conditions and application requirements. Traffic flows can be dynamically steered, load-balanced, or rerouted to optimize network performance and ensure quality of service.

**Interoperability and Vendor Neutrality:**

- SDN promotes interoperability and vendor neutrality by standardizing the communication interfaces between the SDN controller and network devices. This allows organizations to deploy heterogeneous network equipment from different vendors while still benefiting from centralized management and programmability.

**Integration with Network Functions Virtualization (NFV):**

- SDN and NFV are complementary technologies that work together to virtualize and automate network infrastructure. SDN provides the control and management framework.

# **CHAPTER 4**

## **SYSTEM**

## **ANALYSIS**

System requirements are the functionality that is needed by a system in order to satisfy the customer's requirements. System requirements are broad and a narrow subject that could be implemented to many items. The requirements document allows the project team to have a clear picture of what the software solution must do before selecting a vendor. Without an optimized set of future state requirements, the project team has no effective basis to choose The best system for your organization.

### **4.1 HARDWARE AND SOFTWARE REQUIREMENTS**

#### **4.1.1 HARDWARE REQUIREMENTS**

- Processor: Intel Core i3
- RAM : 4GB
- Hard Disk: 100 GB
- Components: Base Station, g node B, Vaman

#### **4.1.2 SOFTWARE REQUIREMENTS**

- Operating System: Ubuntu
- Packages: C/C++, ns3-tools, CMake
- IDE: Visual Studio Code
- Wireshark
- NetAnim

## **4.2 SOFTWARE REQUIREMENTS SPECIFICATION**

The software requirements specification outlines both the functional and non-functional requirements necessary for implementing Wireshark tracing, ASCII tracing, NetAnim visualization, and TCP congestion control analysis functionalities within the ns-3 framework, specifically focusing on 5G network simulations.

### **4.2.1 FUNCTIONAL REQUIREMENTS:**

#### **1) Wireshark Tracing Functionality:**

- **Packet Capture:** Enable capturing of packet-level details, including headers and payloads, during 5G network simulations.
- **Configuration:** Allow users to configure Wireshark tracing parameters such as capture filters, interface selection, and capture duration.
- **Integration:** Seamlessly integrate Wireshark tracing into ns-3 simulations to capture packet traces accurately for analysis.
- **Data Analysis:** Provide tools or guidelines for analyzing captured packet traces using Wireshark's features to troubleshoot network issues and analyze performance.

#### **2) ASCII Tracing Functionality:**

- **Trace Generation:** Enable the generation of trace files containing movement and interaction information among 5G network components in ASCII format.
- **Configuration:** Allow users to configure ASCII tracing parameters such as output file format, directory, and verbosity level.
- **Interpretation:** Provide guidelines or tools for parsing and interpreting ASCII trace files to extract relevant data about 5G network behavior and performance.

#### **3) NetAnim Visualization Functionality:**

- **Animation Generation:** Generate animation XML trace files representing the movement and interactions of nodes, base stations, and user equipment (UE) during 5G network simulations.
- **Visualization:** Facilitate visualization of 5G network simulations using the NetAnim graphical tool, with support for depicting UE mobility, handovers, and data flow.
- **Customization:** Allow users to customize visualization parameters such as node positions, colors, animation speed, and the representation of 5G-specific elements like gNBs and UEs.
- **Real-time Monitoring:** Provide real-time monitoring capabilities to visualize 5G network behavior dynamically during simulation execution, enabling observation of handovers, resource allocations, and traffic patterns.

#### **4) TCP Congestion Control Analysis Functionality:**



- **Congestion Window Tracing:** Trace TCP congestion window changes during 5G network simulations to analyze the performance of congestion control mechanisms.
- **Data Collection:** Collect data on TCP congestion control behavior, including window size dynamics, congestion avoidance, and recovery mechanisms.
- **Analysis Tools:** Provide tools or guidelines for analyzing collected data to evaluate the effectiveness of TCP congestion control algorithms in 5G scenarios and their impact on end-to-end throughput and latency

#### 4.2.2 NON-FUNCTIONAL REQUIREMENTS:

##### 1. Performance:

- **Efficiency:** Ensure that Wireshark tracing, ASCII tracing, NetAnim visualization, and TCP congestion control analysis functionalities do not significantly degrade simulation performance.
- **Scalability:** Support large-scale 5G network simulations with numerous base stations, UEs, and traffic flows without compromising performance or stability.

##### 2. Reliability:

- **Accuracy:** Ensure that captured packet traces, ASCII trace files, and simulation data accurately represent the behavior of 5G networks and TCP congestion control mechanisms.
- **Stability:** Maintain stability and robustness throughout the simulation process to prevent crashes, data corruption, or unexpected behavior.

##### 3. Usability:

- **Ease of Use:** Provide intuitive interfaces, clear documentation, and user-friendly controls to facilitate interaction with Wireshark tracing, ASCII tracing, NetAnim visualization, and TCP congestion control analysis functionalities.
- **Accessibility:** Ensure that functionalities are accessible to users of varying experience levels, from novice to expert, through well-designed interfaces and documentation.

##### 4. Compatibility:

- **Integration:** Ensure seamless integration of Wireshark tracing, ASCII tracing, NetAnim visualization, and TCP congestion control analysis functionalities with the ns-3 framework for 5G network simulations.
- **Platform Independence:** Support multiple operating systems and environments to maximize compatibility and accessibility for users across different platforms.

##### 5. Security:

- **Data Privacy:** Implement measures to protect sensitive data captured during packet tracing and simulation, ensuring privacy and compliance with data protection regulations.
- **Authentication:** Implement authentication mechanisms to prevent unauthorized access to network simulation data, trace files, and visualization tools.

##### 6. Maintainability:

- **Modularity:** Design functionalities in a modular and extensible manner to facilitate maintenance and future enhancements of Wireshark tracing, ASCII tracing, NetAnim

visualization, and TCP congestion control analysis features.

- **Documentation:** Provide comprehensive documentation covering installation, configuration, and usage of the implemented functionalities to support ongoing maintenance, troubleshooting, and development efforts.

# **CHAPTER 5**

## **SYSTEM DESIGN**

### **5.1 DESCRIPTION**

System design is the process of creating a system's architecture, parts, and interfaces to ensure that it satisfies the needs of its users.

Thus, in order to examine the design of this project, we first go through the specifics of establishing the concept of drone detection through a few fundamental modules that would clearly describe the workings of the system that would come from the development.

#### **5G Wireless Communication**

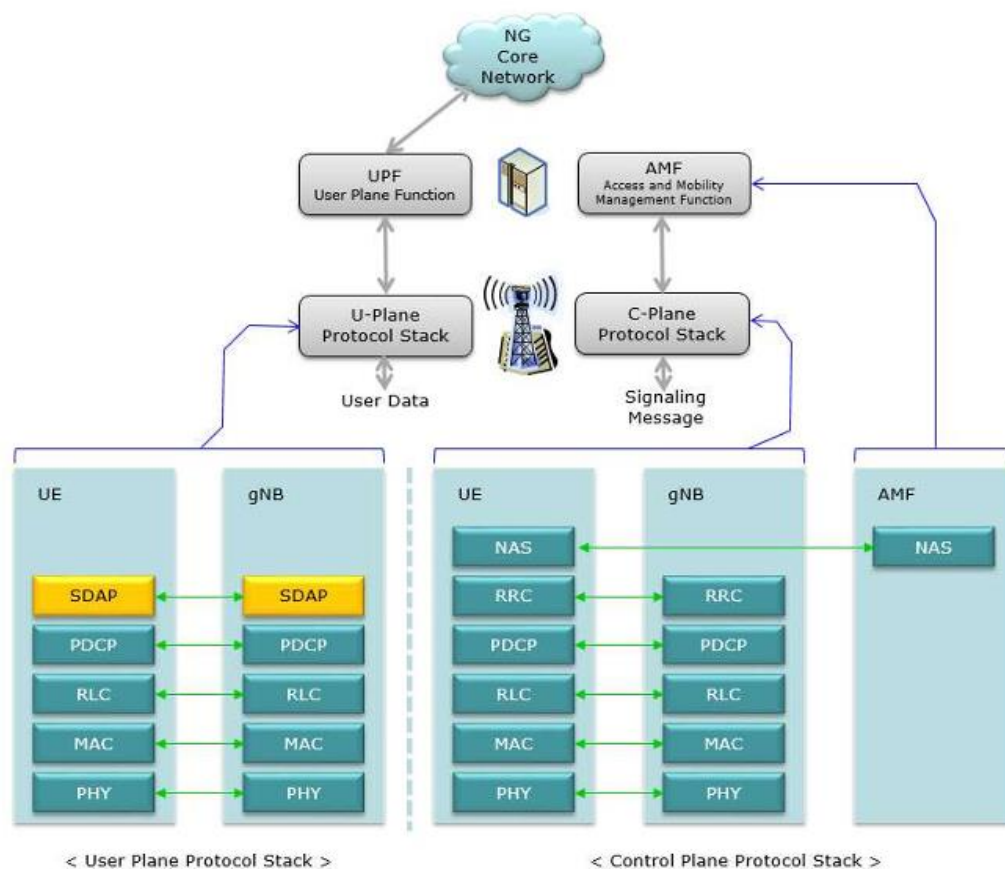
The architecture of 5G networks is meticulously designed to meet the evolving demands of modern communication, promising to revolutionize connectivity across various sectors. At the heart of 5G lies the Radio Access Network (RAN), powered by the innovative New Radio (NR) technology. NR operates in both sub-6 GHz and mmWave frequency bands, offering unprecedented data rates, ultra-low latency, and enhanced spectral efficiency. Base Stations, known as gNBs (Next-Generation NodeB), form the backbone of 5G RAN, leveraging advanced antenna technologies like massive MIMO and beamforming to provide extensive coverage and deliver high-quality wireless connectivity to user devices.

Complementing the RAN is the Core Network (CN), characterized by its Service-Based Architecture (SBA) that facilitates modular deployment of network functions as scalable services. Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) principles are integral to the 5G core, enabling dynamic resource allocation, efficient management, and rapid service deployment. The Core Network comprises essential components such as the User Plane Function (UPF), Session Management Function (SMF), and Authentication and Authorization Function (AAF), ensuring seamless connectivity, mobility management, and secure access for users.

Furthermore, 5G architecture embraces Edge Computing and Mobile Edge Computing (MEC) to bring computation and service execution closer to end-users. MEC platforms deployed at the network edge empower low-latency applications, IoT services, and content caching, unlocking new possibilities for real-time interactions and immersive experiences. Interworking with legacy networks is another

hallmark of 5G architecture, ensuring smooth migration paths and backward compatibility with existing LTE and legacy systems.

In summary, the architecture of 5G networks represents a paradigm shift in telecommunications, offering unparalleled performance, flexibility, and scalability to support a myriad of applications and services. By embracing cutting-edge technologies and standards compliance, 5G architecture sets the stage for a connected world where seamless connectivity, immersive experiences, and transformative innovations become the new norm.



In the context of 5G architecture, the User Plane and Control Plane represent two distinct components responsible for different aspects of network operation and management.

User Plane:

The User Plane, also known as the Data Plane, primarily handles the transmission of user data packets across the network. It focuses on efficient and reliable delivery of data from the source to the destination with minimal latency. Here's a breakdown of key elements within the User Plane:

## 5G Reference Point Architecture

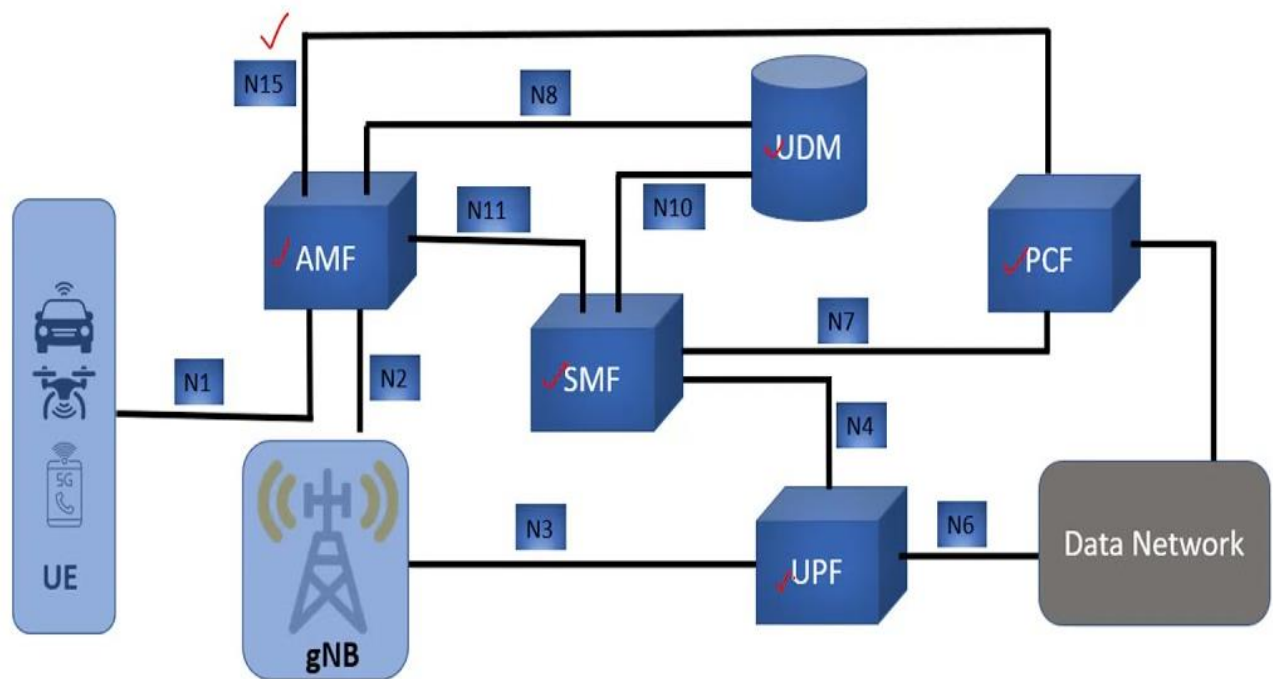


Fig 8: 5G Reference Point Architecture.

1. UPF (User Plane Function): The UPF is a critical component of the User Plane responsible for processing and forwarding user data packets. It performs functions such as packet routing, forwarding, and Quality of Service (QoS) enforcement to ensure optimal data delivery.

2. User Data: User data packets contain information generated or requested by end-user devices (UEs), such as smartphones, IoT devices, or computers. These packets traverse the User Plane, undergoing various processing stages before reaching their destination.

3. U-Plane Protocol Stack: The User Plane Protocol Stack comprises layers of protocols responsible for handling user data packets. These layers typically include the Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC), Medium Access Control (MAC), and Physical (PHY) layer. Each layer performs specific functions to facilitate data transmission and reception.

#### Control Plane:

In contrast, the Control Plane focuses on network control and management functions, including signaling, authentication, and mobility management. It handles the establishment, maintenance, and termination of connections between network elements. Here are the key components of the Control Plane:

1. AMF (Access and Mobility Management Function): The AMF is a vital element of the Control Plane responsible for managing user access and mobility within the network. It handles tasks such as user authentication, mobility management, and session establishment.

2. Signaling Messages: Signaling messages are exchanged between network elements to facilitate control and management functions. These messages carry essential information for establishing connections, managing resources, and maintaining network integrity.

Control Plane Protocol Stack: The Control Plane Protocol Stack comprises protocols responsible for handling signaling and control functions within the network. Protocols such as Non-Access Stratum (NAS), Radio Resource Control (RRC), and Service Data Adaptation Protocol (SDAP) are part of this stack. They facilitate procedures related to mobility management, session establishment, and QoS management.

In summary, while the User Plane focuses on the efficient transmission of user data packets, the Control Plane handles network control and management functions to ensure the smooth operation of the 5G

network. By segregating these functionalities, the architecture optimizes resource utilization, enhances network performance, and enables seamless connectivity for end users.

### **Components in 5g:**

#### **a) UE:**

A UE, or User Equipment, refers to the device used by end-users to access cellular networks such as 5G. It encompasses a wide range of devices including smartphones, tablets, IoT gadgets, and more, capable of connecting to the network infrastructure. UE serves as the interface between users and the network, enabling communication, data exchange, and access to various services and applications. It operates across multiple layers of the protocol stack, from the physical layer responsible for transmitting data over the air interface to the application layer providing user-facing functionalities. UE plays a crucial role in facilitating seamless connectivity, supporting high-speed data transmission, low-latency communication, and diverse use cases ranging from mobile broadband to IoT deployments

## **Layers in UE/ GnodeB :**

### **1. PHY (Physical Layer):**

- The Physical Layer is the lowest layer of the protocol stack and is responsible for transmitting raw binary data over the air interface.
- It handles tasks such as modulation, coding, and transmission of data bits using techniques like OFDMA (Orthogonal Frequency Division Multiple Access) or SC-FDMA (Single Carrier Frequency Division Multiple Access).
- PHY also manages aspects like power control, channel estimation, and beamforming to optimize the wireless link performance.

### **2. MAC (Medium Access Control):**

- MAC is responsible for managing access to the shared radio resources and scheduling the transmission of data between the UE and the network.
- It coordinates the allocation of resources such as time slots, frequency bands, and power levels to multiple UEs within the cell.
- MAC protocol supports functionalities like random access, contention-based access, and hybrid automatic repeat request (HARQ) for error recovery.

### **3. RLC (Radio Link Control):**

- RLC operates above the MAC layer and ensures reliable and ordered delivery of packets over the radio interface.
- It performs segmentation and reassembly of data packets to fit into the size of radio frames and retransmits lost or corrupted packets using Automatic Repeat request (ARQ) mechanisms.
- RLC provides functionalities like sequence numbering, duplicate detection, and error correction to maintain the integrity and continuity of data transmission.

### **4. PDCP (Packet Data Convergence Protocol):**

- PDCP operates above the RLC layer and provides various functionalities such as header compression, encryption, and integrity protection.
- It compresses the IP header to reduce overhead and optimize the transmission efficiency, especially for small data packets.
- PDCP also encrypts the user data and adds integrity protection to secure the communication over the air interface, ensuring confidentiality and data integrity.



## 5. SDAP (Service Data Adaptation Protocol):

- SDAP is a new addition in the 5G protocol stack and is responsible for adapting different types of service data to the requirements of the underlying layers.
- It provides service-specific QoS (Quality of Service) handling and mapping of traffic flows to appropriate QoS bearers.
- SDAP supports functionalities like flow-based QoS control, packet filtering, and mapping of QoS parameters to radio bearers, enabling efficient handling of diverse services with varying requirements.

### a) G Node B:

G node Band base stations are integral components of cellular networks, each serving as a pivotal link between user equipment (UE) and the network infrastructure. While traditional base stations have been utilized in previous generations of mobile networks like 4G LTE, g Node Bs represent the evolution of base station technology specifically tailored for 5G networks. Both g Node Bs and base stations are responsible for establishing and managing the radio interface, facilitating communication between UEs and the core network. They employ advanced radio technologies such as OFDMA (Orthogonal Frequency Division Multiple Access) and MIMO (Multiple-Input Multiple-Output) to optimize spectral efficiency, enhance coverage, and increase data rates. Additionally, both g Node Bs and base stations support features like beamforming, which concentrates radio signals towards specific UEs, improving signal quality and capacity. However, g Node Bs typically offer enhanced capabilities compared to traditional base stations, such as support for network slicing, low latency communication, and cloud-native architectures, making them essential components for delivering the full potential of 5G networks.

## **RRC- RADIO RESOURCE CONTROL**

**RRC** - In the 5G protocol stack, the Radio Resource Control (RRC) is a key protocol that resides at the third layer, known as the RRC layer. The RRC protocol is responsible for the control of the radio resources between the User Equipment (UE) and the Radio Access Network (RAN), which includes the gNodeB (gNB) in the 5G context.

Here's a brief overview of the RRC at level 3 in the 5G protocol stack:

1. **Connection Establishment and Release:** The RRC protocol manages the establishment, maintenance, and release of the connection between the UE and the gNB. This includes procedures like initial access, handover, and connection re-establishment.

2. Radio Bearer Control: RRC controls the setup, modification, and release of radio bearers, which are logical channels used to transfer user and control data between the UE and the gNB.

3. Mobility Procedures: RRC manages mobility-related procedures such as handover, which involves transferring the UE's connection from one gNB to another without interrupting the ongoing services.

4. Quality of Service (QoS) Management: The RRC protocol is responsible for managing and maintaining the QoS parameters, ensuring that the required quality of service is provided to the UE based on its requirements and network conditions.

5. Security: RRC handles the security-related procedures, including authentication, ciphering, and integrity protection, to ensure the confidentiality and integrity of the data transmitted between the UE and the gNB.

6. System Information Broadcast: RRC is responsible for broadcasting system information to the UEs, which includes information about the network configuration, available services, and cell parameters.

7. UE Capability Negotiation: RRC facilitates the negotiation and exchange of capabilities between the UE and the gNB, ensuring that both entities are aware of each other's capabilities and can operate efficiently.

Overall, the RRC protocol plays a crucial role in managing and controlling the radio resources in the 5G network, ensuring efficient and reliable communication between the UE and the gNB while maintaining the required QoS and security levels.

## **NAS – NON ACCESS STRATUM**

1. NAS – Non Access Stratum Registration and Mobility Management: The NAS layer manages the registration of the UE with the core network and handles mobility-related procedures such as tracking area updates, location updates, and roaming.

2. Session Management: NAS is responsible for managing the EPS (Evolved Packet System) sessions, including the activation, modification, and deactivation of the bearers for the data transfer between the

UE and the core network.

3. **Security Management:** The NAS layer handles security-related functions, including authentication, key management, ciphering, and integrity protection, to ensure the confidentiality, integrity, and authenticity of the communication between the UE and the core network.
4. **Paging and Notification:** NAS manages the paging and notification procedures, allowing the core network to contact the UE when there is incoming data or signaling messages.
5. **SMS and Supplementary Services:** NAS supports Short Message Service (SMS) and supplementary services, allowing the UE to send and receive SMS messages and access additional services provided by the core network.
6. **Service Request and Release:** NAS handles the procedures for initiating, modifying, and terminating services requested by the UE, ensuring smooth communication and resource utilization within the network.
- .
7. **Network Selection and Connection Establishment:** The NAS layer facilitates the selection of the appropriate core network and the establishment of connections with the selected network, based on the UE's location, preferences, and network conditions.
8. **User Identity Handling:** NAS manages the handling and protection of the user's identity information, ensuring privacy and security when the UE communicates with the core network.
9. The NAS layer plays a crucial role in managing the signaling and control functions for the core network connectivity and services in the 5G network. It ensures seamless communication, mobility, security, and service provisioning for the User Equipment (UE) within the network.

## 5.3 UML DIAGRAMS

UML Diagrams are classified into different types such as

1. DATA FLOW Diagram
2. STATE CHART Diagram
3. USE CASE Diagram
4. CLASS Diagram
5. SEQUENCE Diagram
6. DEPLOYMENT Diagram

### 1. Data Flow Diagram

A data-flow diagram is a visual representation of how data moves through a system or a process (usually an information system). The data flow diagram also shows the inputs and outputs of each entity as well as the process itself. A data-flow diagram lacks control flow, loops, and decision-making processes. With a flowchart, certain operations based on the data can be depicted. As the project encompasses various components and functionalities related to 5G architecture and ns-3 simulation, the data flow diagram would depict the flow of data and control between different modules and layers of the system. Below is a high-level representation of the data flow within the project:

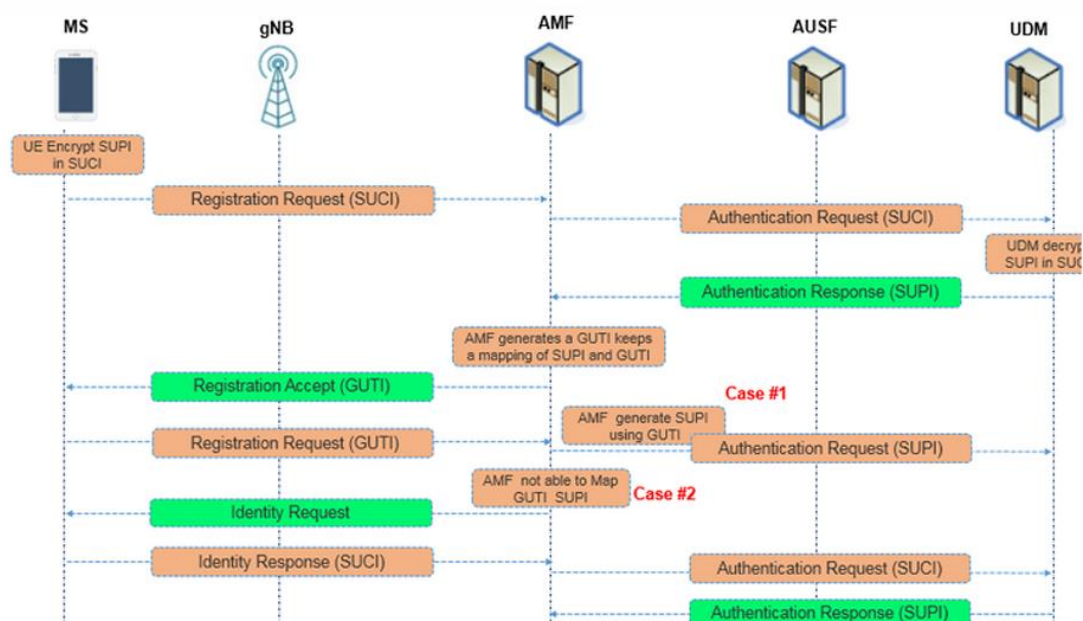


Fig 9: Data Flow.

## 2. Class Diagram

Class diagrams are the main building block of any object-oriented solution. It displays a system's classes, along with each class's properties, operations, and relationships to other classes. Most modelling tools include three elements to a class. Name is at the top, followed by attributes, then operations or methods, and finally, methods. Classes are linked together to generate class diagrams in a complex system with numerous related classes. Various sorts of arrows represent different relationships between classes. Creating a class diagram for the integration of 5G functionalities into the ns-3 framework involves illustrating the key classes and their relationships. Below is a simplified class diagram focusing on the main components involved in enabling Wireshark tracing, ASCII tracing, NetAnim visualization, and TCP congestion control analysis within ns-3 for 5G network simulation and analysis:

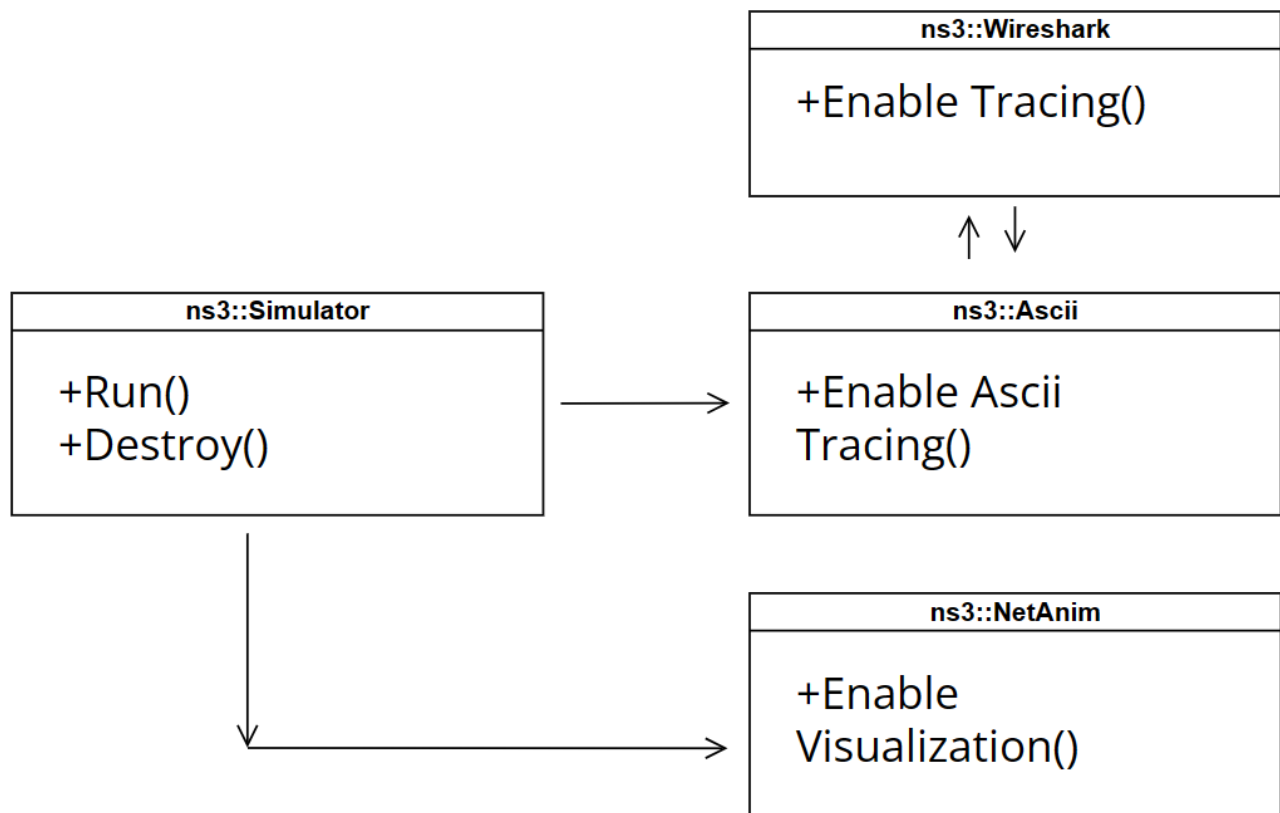


Fig 10: Class Diagram.

### 3. Sequence Diagram

In UML, sequence diagrams display how and in what order certain items interact with one another. It's crucial to remember that they depict the interactions for a certain circumstance. The interactions are depicted as arrows, while the processes are portrayed vertically. Sequence diagram for the entire project involving 5G architecture and ns-3 simulation would be quite complex due to the multitude of interactions and components involved. However, I can provide a simplified sequence diagram illustrating the main interactions between different modules and layers.

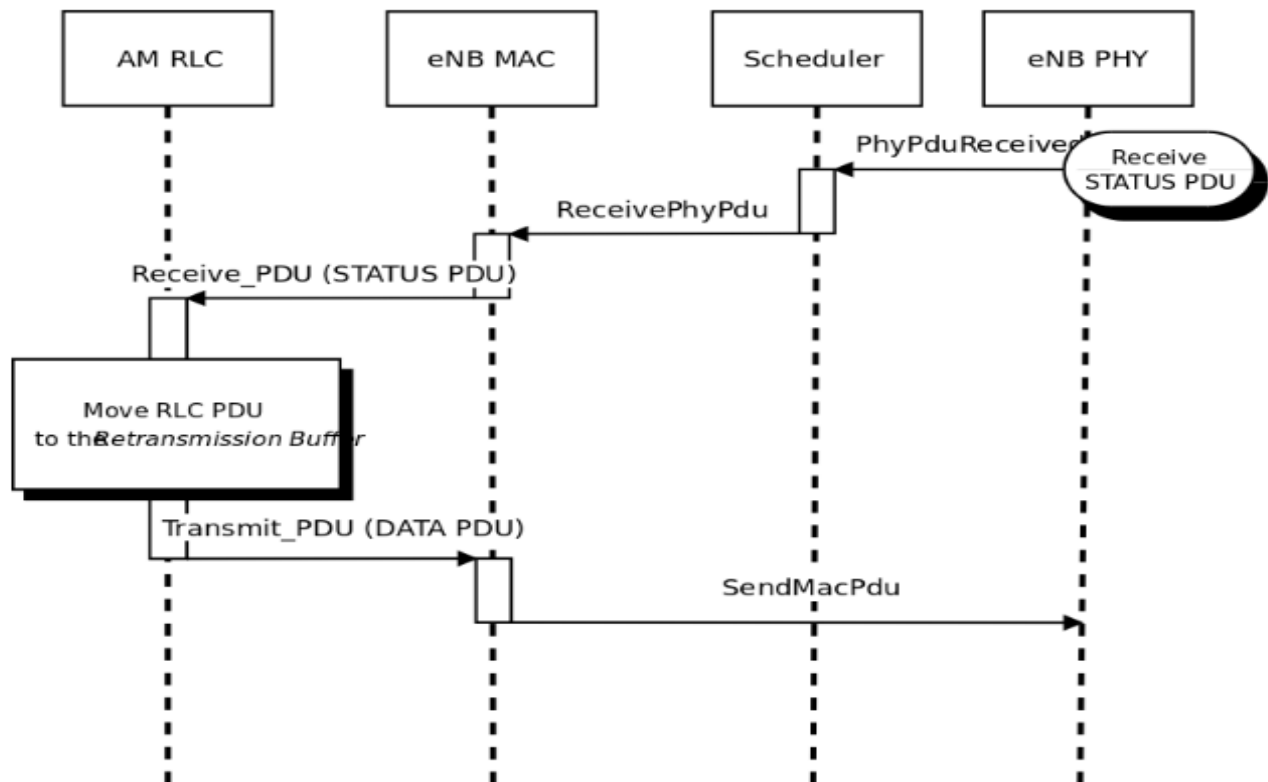


Fig 11: Sequence Diagram.

#### 4. Deployment Diagram

Deployment diagrams are used to show the hardware components of a system, their communication paths, and the locations of software files on that hardware.

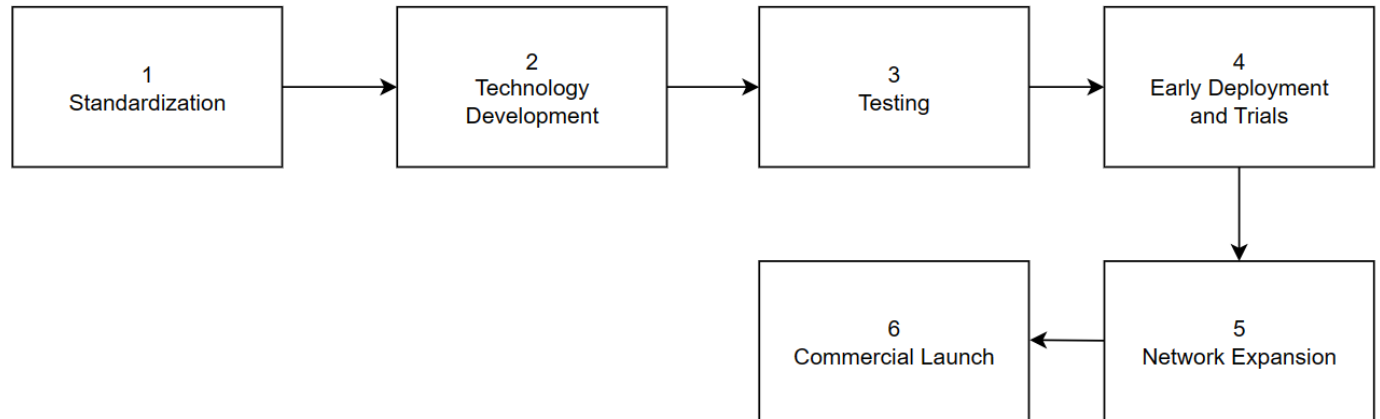


Fig 12: Deployment Diagram.

# CHAPTER 6

## INPUT/OUTPUT

### Key Abstractions in NS-3

#### Node

- **Definition:** Represents a basic computing device in NS-3.
- **Description:** Equivalent to a computer in networking terminology, where additional functionality like applications and protocol stacks can be added.
- **Representation:** C++ class Node.
- **Example:** A computer in a network setup.

#### Application

- **Definition:** Represents a user program generating activity in simulations.
- **Description:** Equivalent to software applications running on computers to perform tasks.
- **Representation:** C++ class Application.
- **Example:** UdpEchoClientApplication and UdpEchoServerApplication for generating and echoing network packets.

#### Channel

- **Definition:** Represents a communication subnetwork in NS-3.
- **Description:** Equivalent to the media over which data flows in real-world networks.
- **Representation:** C++ class Channel.
- **Example:** CsmaChannel for carrier sense multiple access communication medium.

#### Net Device

- **Definition:** Represents both software driver and simulated hardware for network communication.
- **Description:** Equivalent to network interface cards (NICs) in real computers.
- **Representation:** C++ class NetDevice.
- **Example:** CsmaNetDevice, PointToPointNetDevice, and WifiNetDevice.

#### Topology Helpers

- **Definition:** Simplify common tasks like connecting NetDevices to Nodes and Channels.
- **Description:** Combine multiple NS-3 core operations into easy-to-use models.
- **Representation:** Provided as topology helper objects.
- **Example:** Simplifying operations to connect devices and networks in large simulated networks.



## **First.cc:**

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
```

Include Directives: These lines include necessary header files from the ns-3 modules. These headers provide classes, functions, and constants needed for networking simulation.

```
// Default Network Topology
//
//      10.1.1.0
// n0 ----- n1
//      point-to-point
```

Topology Comment: This comment section illustrates the default network topology planned for the simulation.

```
using namespace ns3;
```

```
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
```

Namespace and Logging Component: This line uses the namespace ns3 for convenience and defines a logging component named "FirstScriptExample" using NS\_LOG\_COMPONENT\_DEFINE.

```
int main (int argc, char *argv[])
{
    CommandLine cmd (__FILE__);
    cmd.Parse (argc, argv);
```

Main Function: This is the entry point of the program. It parses command-line arguments using CommandLine to handle any command-line arguments passed to the program.

```
Time::SetResolution (Time::NS);
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

Time Resolution and Logging Configuration: It sets the time resolution to nanoseconds using Time::SetResolution and enables logging for UDP echo client and server applications at the INFO level.

```
NodeContainer nodes;
```

```
nodes.Create (2);
```

Node Creation: It creates a container to hold two nodes for the simulation.

```
PointToPointHelper pointToPoint;  
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));  
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

```
NetDeviceContainer devices;  
devices = pointToPoint.Install (nodes);
```

Point-to-Point Link Configuration: It sets up a point-to-point link between the nodes with a specified data rate and delay using PointToPointHelper. Then, it installs the devices on the nodes.

```
InternetStackHelper stack;  
stack.Install (nodes);
```

```
Ipv4AddressHelper address;  
address.SetBase ("10.1.1.0", "255.255.255.0");
```

```
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Internet Stack and IP Address Assignment: It installs the internet stack on the nodes using InternetStackHelper. Then, it assigns IPv4 addresses to the point-to-point devices.

```
UdpEchoServerHelper echoServer (9);  
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));  
serverApps.Start (Seconds (1.0));  
serverApps.Stop (Seconds (10.0));
```

UDP Echo Server Setup: It sets up a UDP echo server on node 1 with port 9. It starts the server application at 1 second and stops it at 10 seconds.

```
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);  
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));  
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));  
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));
```

```
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));  
clientApps.Start (Seconds (2.0));  
clientApps.Stop (Seconds (10.0));
```

UDP Echo Client Setup: It sets up a UDP echo client on node 0, targeting the address of node 1 with port 9. It specifies the client's attributes such as the number of packets, interval between packets, and packet size. The client application starts at 2 seconds and stops at 10 seconds.

```
Simulator::Run ();  
Simulator::Destroy ();  
return 0;  
}
```

```
•nikhil@hp-pavilion:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/first.cc  
Consolidate compiler generated dependencies of target scratch_first  
At time +2s client sent 1024 bytes to 10.1.1.2 port 9  
At time +2.00284s server received 1024 bytes from 10.1.1.1 port 49153  
At time +2.00284s server sent 1024 bytes to 10.1.1.1 port 49153  
At time +2.00569s client received 1024 bytes from 10.1.1.2 port 9
```

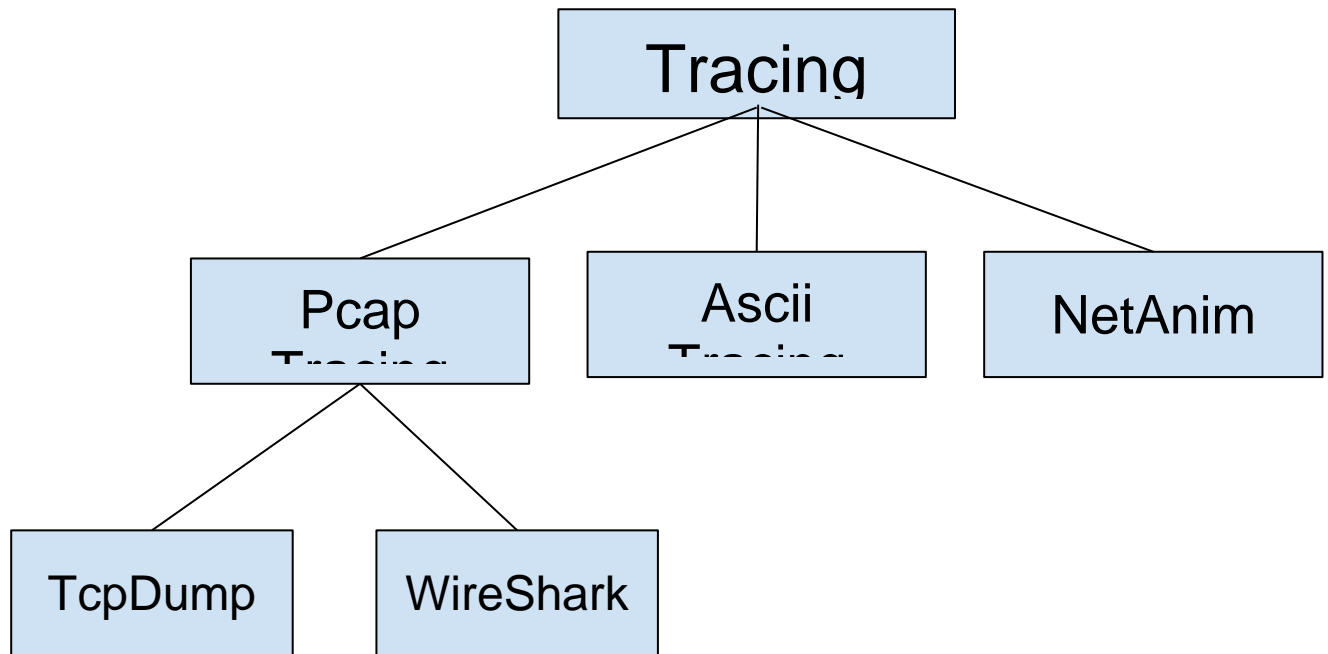
Simulation Execution: It starts the ns-3 simulator and runs the simulation. Finally, it cleans up the simulation environment and exits the program.

This first.cc script sets up a simple network topology with two nodes connected by a point-to-point link, and it runs a UDP echo client-server application over this network. The script demonstrates basic ns-3 usage, including node and device creation, configuration of network parameters, application setup, and simulation.

## CHAPTER 7

### IMPLEMENTATION

#### Tracing in ns-3:



Tracing is a vital mechanism in ns-3 simulations that allows users to capture and analyze data generated during the simulation process. It provides insights into various events and activities occurring within the simulated network environment. This document provides an overview of tracing in ns-3, including its types and usage.

Tracing refers to the process of capturing and recording specific events or actions that occur during the execution of a simulation in ns-3. These events can include packet transmissions, node movements, protocol events, and more.

#### **Types of Tracing:**

##### **a. Packet Tracing:**

- Packet tracing involves capturing information related to packet transmissions within the simulated network.
- It provides details such as packet headers, payloads, source and destination addresses, and timestamps.

- Packet tracing is useful for analyzing network performance, debugging packet routing issues, and evaluating protocol efficiency.

#### **b. Event Tracing:**

- Event tracing involves capturing and recording various events triggered during the simulation.
- Events can include node movements, protocol state changes, link failures, and timer expirations.
- Event tracing helps in understanding the behavior of network protocols, identifying bottlenecks, and diagnosing simulation issues.

#### **c. Flow Tracing:**

- Flow tracing focuses on capturing data related to data flows or communication sessions within the network.
- It provides information about flow initiation, termination, data transfer rates, and end-to-end delay.
- Flow tracing is valuable for analyzing application-level performance, evaluating Quality of Service (QoS), and optimizing network resource allocation.

#### **d. Application Tracing:**

- Application tracing involves capturing events specific to application-layer protocols and functionalities.
- It includes information such as application state changes, message exchanges, and application-specific metrics.
- Application tracing aids in debugging application-level issues, assessing application performance, and optimizing application behavior.

#### **Usage of Tracing:**

- Tracing is typically configured using the ns-3 TraceHelper module, which provides APIs for enabling and configuring tracing mechanisms.
- Tracing can be enabled globally for all nodes and components or selectively for specific nodes, devices, protocols, or events.
- Users can customise tracing parameters such as trace file format, verbosity level, and output location.

- Tracing data is usually stored in trace files in a structured format, which can be later analysed using post-processing tools or scripts.

### **Benefits of Tracing:**

- Provides detailed insights into network behaviour and performance.
- Facilitates debugging and troubleshooting of simulation scenarios.
- Enables performance analysis and optimization of network protocols and applications.
- Supports empirical validation of simulation models against real-world scenarios.

Tracing is a fundamental tool in ns-3 simulations for capturing and analysing runtime data. By enabling various types of tracing mechanisms, users can gain valuable insights into network behaviour, diagnose issues, and optimise simulation models. Understanding the different types of tracing and their usage is essential for effectively leveraging tracing capabilities in ns-3 simulations.

### **PCAP Tracing:**

PCAP tracing is a method used to capture network traffic during ns-3 simulations. It enables users to record packet-level details, including packet headers, payloads, timestamps, and more, for analysis and debugging purposes. PCAP traces can be analyzed using external tools such as tcpdump for Wireshark, providing insights into protocol behavior, packet flows, and network performance.

### **Enabling PCAP Tracing in ns-3**

To enable PCAP tracing in ns-3 simulations, follow these steps:

#### **TCPDUMP Tracing:**

- **Enable PCAP Tracing:** In your ns-3 script, use the `EnablePcapAll` method to enable PCAP tracing for all network devices or `EnablePcap` method to enable tracing for specific devices.
- **Specify Output File:** Specify the output file name for the captured traces.
- **Run Simulation:** Execute the ns-3 simulation script as usual.
- **Analyse Traces with TCPDUMP:** After the simulation, use tcpdump from the command line to analyze the captured traces.

Example code snippet to enable TCPDUMP tracing in ns-3

```
pointToPoint.EnablePcapAll("output");
```

## Wireshark Tracing in ns-3:

Wireshark tracing in ns-3 allows users to capture packet-level details during network simulations for in-depth analysis. This document provides a comprehensive guide to enabling Wireshark tracing in ns-3 simulations, capturing packet traces, and analyzing the captured data using Wireshark.

### Introduction to Wireshark Tracing

Wireshark is a powerful network protocol analyzer that allows users to capture and analyze packet-level details in real-time. Wireshark tracing in ns-3 enables the capture of packet traces during network simulations, providing insights into protocol behavior, packet flows, and network performance.

### Enabling Wireshark Tracing in ns-3

To enable Wireshark tracing in ns-3 simulations, follow these steps:

- **Install Wireshark:** Ensure that Wireshark is installed on your system.
- **Enable PCAP Support:** Configure ns-3 with PCAP support enabled.
- **Enable PCAP Tracing:** In your ns-3 script, enable PCAP tracing for the desired network devices using the `EnablePcapAll` method or `EnablePcap` method for specific devices.
- **Set Output File:** Specify the output file name for the captured traces.

Example code snippet to enable Wireshark tracing in ns-3:

```
// Enable PCAP tracing for all devices
pointToPoint.EnablePcapAll("trace/wireshark_trace");
// Alternatively, enable PCAP tracing for specific devices
phy.EnablePcap("trace/wireshark_trace", wifiDevices.Get(0));
```

### Running the Simulation

Once Wireshark tracing is enabled in your ns-3 script, run the simulation as usual using the ns-3 simulator.

### Capturing Packet Traces

While the ns-3 simulation is running, Wireshark will capture packet traces in real-time according to the configured tracing settings. These traces will be saved to the specified output file location.

## Analysis of the Captured Traces with Wireshark

After the simulation is complete, open Wireshark and load the captured trace file for analysis:

- Launch Wireshark .
- Go to File > Open and select the captured trace file.
- Wireshark will load the trace file, allowing you to analyze packet details, protocols, conversations, and more.

## Wireshark Analysis Features

Wireshark provides a wide range of analysis features, including:

- **Packet Details:** View detailed information about each captured packet, including headers, payloads, and timestamps.
- **Protocol Hierarchy:** Analyze the protocol stack and identify protocols used in the captured traffic.
- **Conversations:** Explore network conversations between hosts and protocols.
- **Filtering:** Apply display filters to focus on specific packets or protocols.
- **Statistics:** Generate statistical summaries and charts based on captured data.

## Best Practices

- **Selective Tracing:** Capture traces selectively to avoid excessive file sizes.
- **Filtering:** Use Wireshark's filtering capabilities to focus on relevant packets and protocols.
- **Analysis Profiles:** Create custom analysis profiles in Wireshark for different types of simulations or analysis scenarios.
- **Collaboration:** Share captured trace files with collaborators for joint analysis and debugging.

Wireshark tracing in ns-3 provides a powerful mechanism for capturing and analyzing packet-level details during network simulations. By following this guide, users can enable Wireshark tracing, capture packet traces, and analyze the captured data effectively using Wireshark's extensive features and capabilities.

## ASCII Tracing in ns-3

ASCII tracing in ns-3 provides a straightforward way to generate trace files containing packet



movement information in ASCII format. These trace files can be useful for debugging, analysis, and visualization purposes. This document outlines how to enable ASCII tracing in ns-3, parse the generated trace files, and interpret the traced events.

## Enabling ASCII Tracing

To enable ASCII tracing in ns-3, you can use the `AsciiTraceHelper` class along with appropriate trace sink methods. Here's how to enable ASCII tracing in a simulation script:

```
AsciiTraceHelper ascii;  
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("myfirst.tr"));
```

In this code snippet:

- `AsciiTraceHelper` is used to create an instance for handling ASCII traces.
- `EnableAsciiAll()` method is called on the relevant network devices (e.g., point-to-point devices) to enable ASCII tracing for them.
- `CreateFileStream()` method is used to create a file stream object representing the trace file ("myfirst.tr").

## Parsing Ascii Traces

After running the simulation and generating the trace file, you can parse the ASCII trace file to analyze the traced events. Each line in the trace file corresponds to a trace event and contains information about packet movement. Here's a breakdown of the trace event format:

- `+`: An enqueue operation occurred on the device queue.
- `-`: A dequeue operation occurred on the device queue.
- `d`: A packet was dropped, typically due to queue overflow.
- `r`: A packet was received by the net device.

The trace file also includes information such as simulation time, trace source namespace, packet headers, and payload details. You can analyze these details to understand the packet flow and network behavior during the simulation.

## Example Trace Event

Here's an example of a trace event from an ASCII trace file: `+`

2.25732

/NodeList/1/DeviceList/0/\$ns3::PointToPointNetDevice/MacRx

ns3::Ipv4Header (

tos 0x0 ttl 64 id 0 protocol 17 offset 0 flags [none]

length: 1052 10.1.1.1 > 10.1.1.2)

ns3::UdpHeader (

length: 1032 49153 > 9)

Payload (size=1024)

In this event:

- +: Indicates an enqueue operation.
- 2.25732: Simulation time.
- /NodeList/1/DeviceList/0/\$ns3::PointToPointNetDevice/MacRx: Trace source namespace.
- Packet headers and payload details.

ASCII tracing in ns-3 provides a convenient way to capture and analyze packet movement during simulations. By enabling ASCII tracing and parsing the generated trace files, you can gain insights into network behavior and diagnose issues effectively.

## **NetAnim:**

NetAnim is a graphical user interface (GUI) tool used to visualize network simulations conducted using the ns-3 network simulator. It provides a visual representation of the simulated network topology, node movements, packet transmissions, and other network events.

### **Installations:**

#### **Installation on Debian/Ubuntu Linux distribution:**

```
sudo apt-get install mercurial  
sudo apt-get install qt5-default
```

### **Downloading NetAnim:**

```
hg clone http://code.nsnam.org/netanim
```

### **Building and Starting NetAnim:**

```
cd netanim  
qmake NetAnim.pro  
make  
./NetAnim
```

## **Using NetAnim**

Using NetAnim involves two main steps:

1. Generating the animation XML trace file during simulation using ns3::AnimationInterface in the ns-3 code base.
2. Loading the XML trace file generated in Step 1 with the NetAnim application.

## Step 1: Generating XML Trace File

To generate the XML trace file, follow these steps:

1. Ensure that your wscript includes the "netanim" module.
2. Include the header `#include "ns3/netanim-module.h"` in your test program.
3. Add the statement `AnimationInterface anim ("animation.xml");` before `Simulator::Run()`.
4. Optionally, customize the animation parameters such as node positions, mobility models, etc.

## Step 2: Loading XML Trace File with NetAnim

1. Run NetAnim by executing `./NetAnim`.
2. Click on the file-open button and select the XML trace file generated in Step 1.

Below is a sample ns-3 script demonstrating the use of NetAnim:

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;

int main(int argc, char* argv[]) {
    // Create nodes, devices, and channels

    // Setup point-to-point connection

    // Setup NetAnim visualization
    AnimationInterface anim("simulation.xml");
    anim.SetConstantPosition(node1, x1, y1);
    anim.SetConstantPosition(node2, x2, y2);

    Simulator::Run();
    Simulator::Destroy();

    return 0;
}
```

## Enabling Logging in ns-3:

### 1. Understanding Logging Levels:

Before enabling logging, it's important to understand the different logging levels available in ns-3:

LOG\_ERROR

LOG\_WARN  
LOG\_DEBUG  
LOG\_INFO  
LOG\_FUNCTION  
LOG\_LOGIC  
LOG\_ALL

## **2. Setting Logging Levels:**

Logging can be enabled globally or on a component-by-component basis.

Logging levels can be set using environment variables or logging system function calls.

## **3. Using Environment Variables:**

Use the NS\_LOG environment variable to set logging levels.

Example: `$ export NS_LOG=UdpEchoClientApplication=level_all`

## **4. Viewing Log Output:**

Run the ns-3 script as usual.

Log output will include messages from enabled logging components.

## **5. Adding Custom Logging:**

Define a logging component using `NS_LOG_COMPONENT_DEFINE("ComponentName")`.

Add logging messages using macros like `NS_LOG_INFO`, `NS_LOG_DEBUG`, etc.

Example: `NS_LOG_INFO("Creating Topology");`

## **6. Running the Script:**

Build the script using ns3.

Clear the NS\_LOG variable to turn off previous logging configurations.

Example: `$ ./ns3 && export NS_LOG=""`

## **7. Enabling Custom Logging:**

Set the desired logging level for the custom component using the NS\_LOG environment variable.

Example: `$ export NS_LOG=ComponentName=info`

## **8. Running the Script Again:**

Execute the script again to see the new logging messages.

Example: `$ ./ns3 run scratch/myfirst`

This process allows for detailed logging in ns-3 simulations, enabling better debugging and understanding of program flow.

## CHAPTER 8

### TESTING

#### Second.cc:

```
// Default Network Topology
//
// 10.1.1.0
// n0 ----- n1  n2  n3  n4
// point-to-point |  |  |  |
//               =====
//               LAN 10.1.2.0
```

#### Header Includes:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
```

These lines include the necessary header files from the ns-3 module, providing access to various functionalities required for network simulation.

#### Namespace:

```
using namespace ns3;
```

This line introduces the ns3 namespace, which encapsulates classes, functions, and objects provided by the ns-3 simulator.

#### Logging:

```
NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
```

This line defines a logging component named "SecondScriptExample", allowing for logging messages at different verbosity levels for debugging purposes.

#### Main Function:

```
int main (int argc, char *argv[])
{
    // Code goes here
    return 0;
}
```

This is the entry point of the program where execution begins. It takes command-line arguments `argc` and `argv[]`.

### **Command Line Parsing:**

```
bool verbose = true;
uint32_t nCsmas = 3;

CommandLine cmd (__FILE__);
cmd.AddValue ("nCsmas", "Number of \"extra\" CSMA nodes/devices", nCsmas);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

cmd.Parse (argc,argv);
```

This section sets up command-line argument parsing using the `CommandLine` class. It defines two parameters, `nCsmas` and `verbose`, and parses them from command-line arguments.

### **Logging Configuration:**

```
if (verbose)
{
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
```

If the `verbose` flag is set to true, logging for the `UdpEchoClientApplication` and `UdpEchoServerApplication` components is enabled at the INFO level.

### **Network Topology Setup:**

```
NodeContainer p2pNodes;
p2pNodes.Create (2);

NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsmas);
```

This section sets up the network topology by creating two node containers: `p2pNodes` for the point-to-point link and `csmaNodes` for the CSMA LAN.

### **Device Configuration:**

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
```

```
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560))));
```

```
NetDeviceContainer csmaDevices;
```

```
csmaDevices = csma.Install (csmaNodes);
```

This part configures the attributes of the point-to-point and CSMA devices, such as data rate and delay, and installs them on the respective nodes.

### **Internet Stack Configuration:**

```
InternetStackHelper stack;
```

```
stack.Install (p2pNodes.Get (0));
```

```
stack.Install (csmaNodes);
```

It installs the Internet stack on the point-to-point node and all CSMA nodes.

### **IP Address Assignment:**

```
Ipv4AddressHelper address;
```

```
address.SetBase ("10.1.1.0", "255.255.255.0");
```

```
Ipv4InterfaceContainer p2pInterfaces;
```

```
p2pInterfaces = address.Assign (p2pDevices);
```

```
address.SetBase ("10.1.2.0", "255.255.255.0");
```

```
Ipv4InterfaceContainer csmaInterfaces;
```

```
csmaInterfaces = address.Assign (csmaDevices);
```

- his section assigns IP addresses to the point-to-point and CSMA interfaces.

### **Application Setup:**

```
UdpEchoServerHelper echoServer (9);
```

```
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsm));
```

```
serverApps.Start (Seconds (1.0));
```

```
serverApps.Stop (Seconds (10.0));
```

```
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsm), 9);
```

```
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
```

```
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
```

```
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));
```

```
ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
```

```
clientApps.Start (Seconds (2.0));
```

```
clientApps.Stop (Seconds (10.0));
```

This part sets up UDP echo server and client applications and installs them on the CSMA node and point-to-point node, respectively.

### **Routing Table Population:**

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

It populates routing tables to enable proper packet forwarding in the simulated network.



### Packet Capture (PCAP) Setup:

```
pointToPoint.EnablePcapAll ("second");
```

```
csma.EnablePcap ("second", csmaDevices.Get (1), true);
```

This section enables packet capture for all devices in the point-to-point link and only for the second CSMA device.

### Animation Interface:

```
AnimationInterface anim ("second.xml");
```

```
anim.SetConstantPosition (csmaNodes.Get(1), 6, 10);
```

```
anim.SetConstantPosition (csmaNodes.Get(2), 9, 10);
```

```
anim.SetConstantPosition (csmaNodes.Get(3), 12, 10);
```

It creates an animation interface using NetAnim and sets constant positions for specific CSMA nodes to define their positions in the animation.

### Simulation Execution:

```
Simulator::Run ();
```

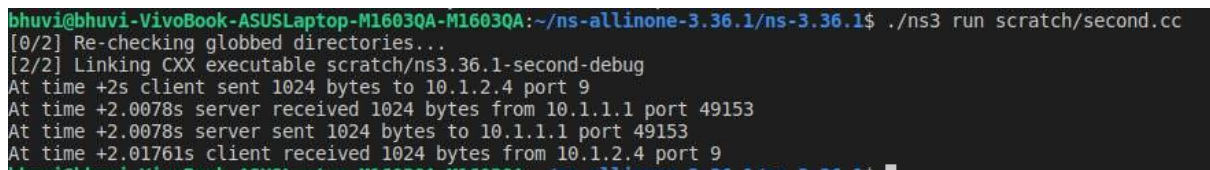
```
Simulator::Destroy ();
```

This part executes the simulation using Simulator::Run() and cleans up resources using Simulator::Destroy() once the simulation completes.

### Return Statement:

```
return 0;
```

Output: Running second.cc file



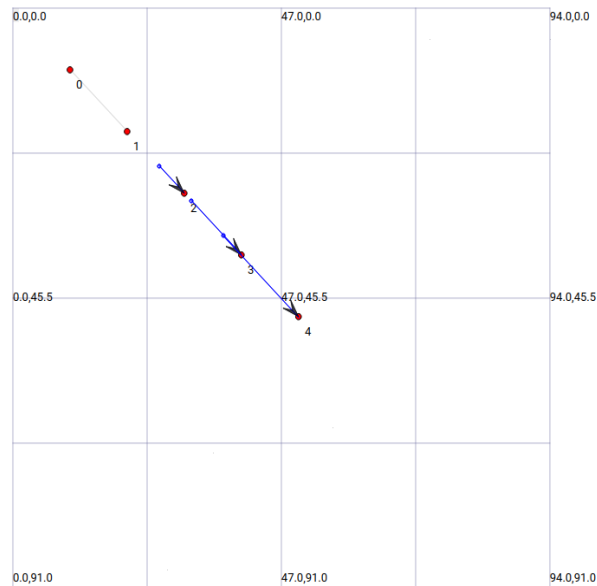
```
bhuvi@bhuvi-VivoBook-ASUSLaptop-M1603QA-M1603QA:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/second.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable scratch/ns3.36.1-second-debug
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.01761s client received 1024 bytes from 10.1.2.4 port 9
```

Running second.xml using NetAnim

The main function returns 0 upon successful execution of the program.

This breakdown explains how the provided code sets up a network simulation using ns-3, configuring network topology, applications, IP addressing, packet capture, and animation.

Also, the packet loss integrated with optimal design of how the data flow is happening by skipping process of encryption and letting PDCP layer to act accordingly to predict and process the directionality of UE by HO process using some of the GNN frameworks those are the part of Deep Learning topology are anticipated and predicted with ARQ.



## Third.cc:

Wireless Network Topology Setup in ns-3

### 1. Introduction

We'll explore the setup of a wireless network topology using the ns-3 simulation framework.

// **Default Network Topology**

//

// **Wifi 10.1.3.0**

// **AP**

// \* \* \* \*

// | | | | **10.1.1.0**

// **n5 n6 n7 n0 ----- n1 n2 n3 n4**

// **point-to-point** | | | |

// =====

// **LAN 10.1.2.0**

### 2. Required Includes

```
#include "ns3/core-module.h"
```

```
#include "ns3/point-to-point-module.h"
```

```
#include "ns3/network-module.h"
```

```
#include "ns3/applications-module.h"
```

```
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
```

These lines include the necessary header files from the ns-3 library, providing functionalities related to core networking, point-to-point communication, network topology, applications, mobility, CSMA, Wi-Fi, etc.

### 3. Network Topology Setup

```
using namespace ns3;
NS_LOG_COMPONENT_DEFINE("ThirdScriptExample");
```

```
int main(int argc, char* argv[]) {
```

The using namespace ns3; statement allows using symbols from the ns3 namespace without explicit qualification.

NS\_LOG\_COMPONENT\_DEFINE("ThirdScriptExample"); defines the logging component for the script.

int main(int argc, char\* argv[]) { marks the entry point of the program, accepting command-line arguments argc and argv.

### 4. Command-Line Parameter Parsing

```
bool verbose = true;
uint32_t nCsmas = 3;
uint32_t nWifi = 3;
CommandLine cmd;
cmd.AddValue("nCsmas", "Number of \"extra\" CSMA nodes/devices", nCsmas);
cmd.AddValue("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);
cmd.Parse(argc,argv);
```

This section sets up command-line parameter parsing, allowing the user to specify the number of CSMA and Wi-Fi nodes and enable verbose logging.

### 5. Network Setup - Creation of Nodes and Channels

```
// Point-to-Point setup
NodeContainer p2pNodes;
p2pNodes.Create(2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install(p2pNodes);
```

```

// CSMA setup
NodeContainer csmaNodes;
csmaNodes.Add(p2pNodes.Get(1));
csmaNodes.Create(nCsmas);
CsmasHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install(csmaNodes);

// Wi-Fi setup
NodeContainer wifiStaNodes;
wifiStaNodes.Create(nWifi);
NodeContainer wifiApNode = p2pNodes.Get(0);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy;
phy.SetChannel(channel.Create());
WifiMacHelper mac;
Ssid ssid = Ssid("ns-3-ssid");
WifiHelper wifi;
NetDeviceContainer staDevices;
mac.SetType("ns3::StaWifiMac",
            "Ssid", SsidValue(ssid),
            "ActiveProbing", BooleanValue(false));
staDevices = wifi.Install(phy, mac, wifiStaNodes);
mac.SetType("ns3::ApWifiMac",
            "Ssid", SsidValue(ssid));
NetDeviceContainer apDevices;
apDevices = wifi.Install(phy, mac, wifiApNode);

```

This section sets up the network topology by creating nodes and channels for point-to-point, CSMA, and Wi-Fi networks. It includes setting device attributes, installing devices on nodes, and configuring Wi-Fi MAC parameters.

## 6. Mobility Setup

```

MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                             "MinX", DoubleValue(0.0),
                             "MinY", DoubleValue(0.0),
                             "DeltaX", DoubleValue(5.0),
                             "DeltaY", DoubleValue(10.0),
                             "GridWidth", UIntegerValue(3),
                             "LayoutType", StringValue("RowFirst"));
mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
                          "Bounds", RectangleValue(Rectangle(-50, 50, -50, 50)));
mobility.Install(wifiStaNodes);
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);

```

This section sets up mobility for the Wi-Fi nodes, defining their initial positions and movement behavior. Wi-Fi stations move randomly within a specified bounding box, while the access point remains stationary.

## 7. Protocol Stack Installation

```
InternetStackHelper stack;  
stack.Install(csmaNodes);  
stack.Install(wifiApNode);  
stack.Install(wifiStaNodes);
```

Here, the Internet protocol stack is installed on all nodes in the CSMA and Wi-Fi networks.

## 8. IP Address Assignment

```
Ipv4AddressHelper address;  
address.SetBase("10.1.1.0", "255.255.255.0");  
Ipv4InterfaceContainer p2pInterfaces;  
p2pInterfaces = address.Assign(p2pDevices);  
address.SetBase("10.1.2.0", "255.255.255.0");  
Ipv4InterfaceContainer csmaInterfaces;  
csmaInterfaces = address.Assign(csmaDevices);  
address.SetBase("10.1.3.0", "255.255.255.0");  
address.Assign(staDevices);  
address.Assign(apDevices);
```

IP addresses are assigned to the devices/interfaces of the point-to-point, CSMA, and Wi-Fi networks.

## 9. Application Setup - Echo Server and Client

```
UdpEchoServerHelper echoServer(9);  
ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));  
serverApps.Start(Seconds(1.0));  
serverApps.Stop(Seconds(10.0));
```

```
UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma), 9);  
echoClient.SetAttribute("MaxPackets", UIntegerValue(1));  
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));  
echoClient.SetAttribute("PacketSize", UIntegerValue(1024));  
ApplicationContainer clientApps =  
    echoClient.Install(wifiStaNodes.Get(nWifi - 1));  
clientApps.Start(Seconds(2.0));  
clientApps.Stop(Seconds(10.0));
```

This section sets up echo server and client applications. The server is installed on the last CSMA node, while the client is installed on the last Wi-Fi station node.

## 10. IP Routing Configuration

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

This line configures IP routing, populating the routing tables with necessary information for packet forwarding.

## 11. Stopping the Simulation

```
Simulator::Stop(Seconds(10.0));
```

This command stops the simulation after 10 seconds of simulated time.

## 12. Tracing Setup

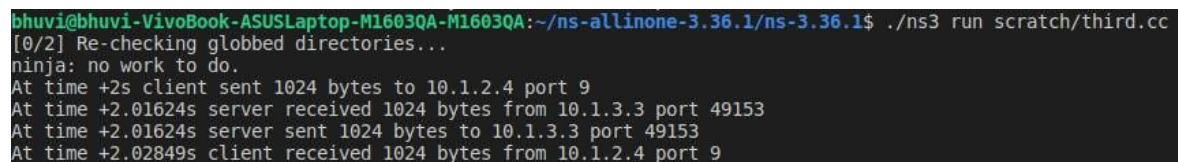
```
pointToPoint.EnablePcapAll("thirdp/third");  
phy.EnablePcap("thirdp/third", apDevices.Get(0));  
csma.EnablePcap("thirdp/third", csmaDevices.Get(0), true);
```

Tracing is enabled to capture packet traces using pcap. Traces are saved to files in the "thirdp" directory. Point-to-point, Wi-Fi AP, and CSMA nodes' traces are enabled.

## 13. Running the Simulation

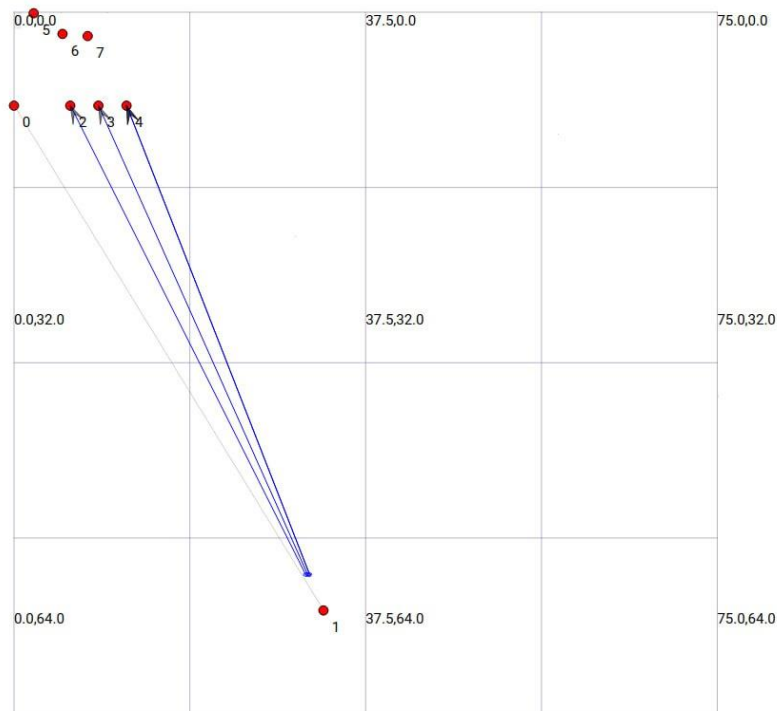
```
Simulator::Run();  
Simulator::Destroy();  
return 0;
```

Output: Running third.cc file



```
bhuvi@bhuvi-VivoBook-ASUSLaptop-M1603QA-M1603QA:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/third.cc  
[0/2] Re-checking globbed directories...  
ninja: no work to do.  
At time +2s client sent 1024 bytes to 10.1.2.4 port 9  
At time +2.01624s server received 1024 bytes from 10.1.3.3 port 49153  
At time +2.01624s server sent 1024 bytes to 10.1.3.3 port 49153  
At time +2.02849s client received 1024 bytes from 10.1.2.4 port 9
```

Running third.xml file Using NetAnim



Finally, the simulation is executed using `Simulator::Run()`, and after completion, resources are cleaned up using `Simulator::Destroy()`.

## **Queues in ns-3**

In ns-3, the choice of queueing disciplines can significantly impact performance, making it essential for users to comprehend the default settings and how to modify them for optimal performance. The architecture of ns-3 separates the device layer from the IP layers or traffic control layers of an Internet host. Outgoing packets in recent ns-3 releases traverse two queueing layers before reaching the channel object: the traffic control layer and the device-specific queueing layer.

### **1. Traffic Control Layer**

At the traffic control layer, active queue management (RFC7567) and prioritization due to quality-of-service (QoS) occur in a device-independent manner through queueing disciplines. Flow control is essential to notify the traffic control layer when the device queue is full, ensuring the effectiveness of

the traffic control layer. Currently, flow control is supported by specific NetDevices.

## 2. Device-Specific Queues

Different NetDevices (e.g., Point-To-Point, Csma, Wi-Fi) have different implementations of queues. By default, queues in ns-3 are not autotuned for link properties but can be dynamically adjusted by enabling Byte Queue Limits (BQL).

### Available Queueing Models in ns-3

- PFifoFastQueueDisc: Default maximum size: 1000 packets
- FifoQueueDisc: Default maximum size: 1000 packets
- RedQueueDisc: Default maximum size: 25 packets
- CoDelQueueDisc: Default maximum size: 1500 kilobytes
- FqCoDelQueueDisc: Default maximum size: 10240 packets
- PieQueueDisc: Default maximum size: 25 packets
- MqQueueDisc: No limits on capacity
- TbfQueueDisc: Default maximum size: 1000 packets

### Changing from Defaults

- Modify the type of queue used by a NetDevice through the device helper.
- Modify the type of queue disc installed on a NetDevice through the traffic control helper.
- Enable BQL on a device that supports it through the traffic control helper.

### Sample Code for Modifying Queues

```
// Setting queue type for NetDevice
PointToPointHelper p2p;
p2p.SetQueue("ns3::DropTailQueue", "MaxSize", StringValue("50p"));

// Modifying queue disc type
TrafficControlHelper tch;
tch.SetRootQueueDisc("ns3::CoDelQueueDisc", "MaxSize", StringValue("1000p"));

// Enabling BQL
tch.SetQueueLimits("ns3::DynamicQueueLimits", "HoldTime", StringValue("4ms"));
```

## **Fourth.cc:**

### Understanding Trace Sources and Trace Sinks

Tracing in ns-3 allows developers to monitor and record the behavior of network simulations by



capturing events and data changes. One fundamental aspect of tracing is understanding trace sources and trace sinks. In this guide, we'll explore how to define trace sources, connect them to trace sinks, and observe the resulting behavior.

## Trace Sources

A trace source in ns-3 represents a point in the code where events or data changes occur and can be observed. It serves as the origin of traceable events and provides hooks for connecting to trace sinks. To define a trace source, follow these steps:

1. **Include Necessary Headers:** Include the required headers for working with tracing, such as `ns3/object.h`, `ns3/traced-value.h`, and `ns3/trace-source-accessor.h`.
2. **Declare the Object:** Create a class that inherits from `ns3::Object`. This class will host the trace source.
3. **Define the Trace Source:** Within the class definition, use the `AddTraceSource` method to declare the trace source. Provide a name, help string, and accessor function for the traceable value.
4. **Use TracedValue:** Declare a member variable of type `ns3::TracedValue<T>` to represent the value that will be traced.

Here's an example of defining a trace source in ns-3:

```
class MyObject : public ns3::Object
{
public:
    static ns3::TypeId GetTypeId()
    {
        static ns3::TypeId tid = ns3::TypeId("MyObject")
            .SetParent(ns3::Object::GetTypeId())
            .AddConstructor<MyObject>()
            .AddTraceSource("MyInteger",
                "An integer value to trace.",
                ns3::MakeTraceSourceAccessor(&MyObject::m_myInt),
                "ns3::TracedValueCallback::Int32");
        return tid;
    }

    MyObject() {}
    ns3::TracedValue<int32_t> m_myInt;
};
```

## Trace Sinks

A trace sink is a function that receives notifications from trace sources when events occur or data changes. These functions, also known as callback functions, process the traced data and perform any desired actions. To define a trace sink:

**Declare the Callback Function:** Define a function with the appropriate signature to serve as the trace sink.

**Connect the Trace Source to the Sink:** Use the `TraceConnectWithoutContext` method to establish a connection between the trace source and the trace sink. Provide the name of the trace source and the callback function.

Here's an example of defining a trace sink and connecting it to a trace source:

```
void IntTrace(int32_t oldValue, int32_t newValue)
{
    std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}

int main(int argc, char *argv[])
{
    ns3::Ptr<MyObject> myObject = ns3::CreateObject<MyObject>();
    myObject->TraceConnectWithoutContext("MyInteger", ns3::MakeCallback(&IntTrace));

    myObject->m_myInt = 1234;
}
```

Tracing in ns-3 involves defining trace sources to capture events or data changes and connecting them to trace sinks, which process the traced data. By understanding how to define trace sources and trace sinks, developers can effectively monitor and analyze network simulations in ns-3.

## **Fifth.cc:**

The `fifth.cc` example program demonstrates the use of ns-3 to analyze the behavior of TCP congestion window under varying network conditions. It involves creating a simple application to send packets over a point-to-point network, tracing changes in the congestion window, and introducing errors to observe their impact on the congestion control mechanism.

The code consists of several sections:

License Header and Includes:

The program begins with the GNU General Public License header and necessary include statements

for ns-3 modules and classes.

```
#include "tutorial-app.h"
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
```

### **Network Illustration and Problem Statement:**

- This section provides a network illustration and addresses the problem of tracing congestion window changes due to limitations in accessing socket information.

### **Custom Application: TutorialApp Class:**

- Defines a custom application TutorialApp responsible for sending packets.
- Implements methods for setting up the socket, starting and stopping the application, scheduling packet transmissions, and sending packets.

```
class TutorialApp : public Application {
    // Class definition and methods
};
```

#### **1. Starting/Stopping Applications:**

- Explanation of how events are started and stopped in ns-3, detailing the scheduling mechanism for application events.

#### **2. Trace Sinks:**

- Implementation of trace sinks for monitoring congestion window changes (CwndChange) and packet drops (RxDrop).

#### **3. Main Program:**

- Configuration of TCP parameters and command-line argument processing.
- Creation of a point-to-point network with error models.
- Installation of TCP receiver (PacketSink) on the destination node.
- Creation of a TCP sender socket and connection to trace source for congestion window tracing.
- Manual instantiation and installation of the TutorialApp application on the source node.
- Connection of trace source for packet drops to the RxDrop trace sink.
- Simulation run and cleanup.

## Code Execution:

The program can be executed using the following command:

```
$ ./ns3 run fifth > cwnd.dat 2>&1
```

This command runs the simulation and redirects the output to a file named cwnd.dat for further analysis.

## Visualization:

The traced data can be visualized using plotting tools like gnuplot. Below is an example command to generate a plot:

```
$ gnuplot
gnuplot> set terminal png size 640,480
gnuplot> set output "cwnd.png"
gnuplot> plot "cwnd.dat" using 1:2 title 'Congestion Window' with linespoints
gnuplot> exit
```

This command generates a PNG plot (cwnd.png) depicting the congestion window behavior over time.

The fifth.cc example program illustrates how ns-3 can be used to analyze TCP congestion control behavior in simulated networks. It demonstrates the creation of custom applications, tracing mechanisms, and visualization techniques to gain insights into network performance.

## OUTPUTS:

First.cc

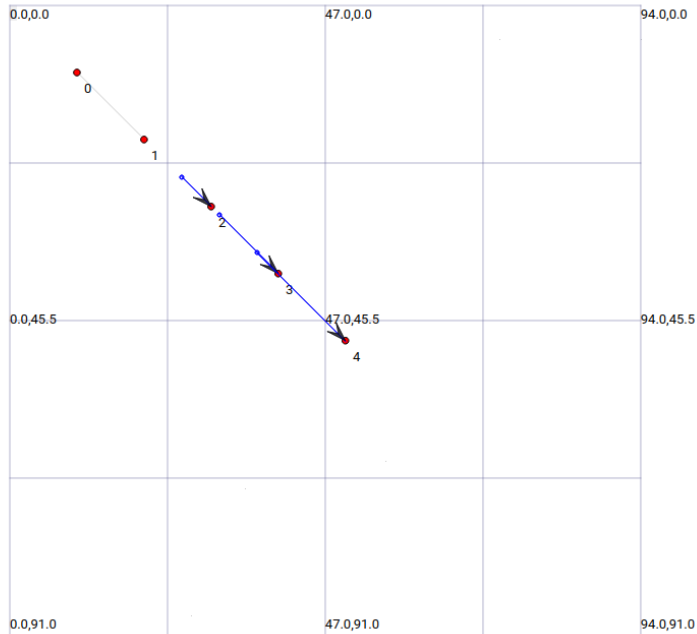
```
•nikhil@hp-pavilion:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/first.cc
Consolidate compiler generated dependencies of target scratch_first
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00284s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00284s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00569s client received 1024 bytes from 10.1.1.2 port 9
```

second.cc

```

bhuvi@bhuvi-VivoBook-ASUSLaptop-M1603QA-M1603QA:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/second.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable scratch/ns3.36.1-second-debug
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.01761s client received 1024 bytes from 10.1.2.4 port 9

```

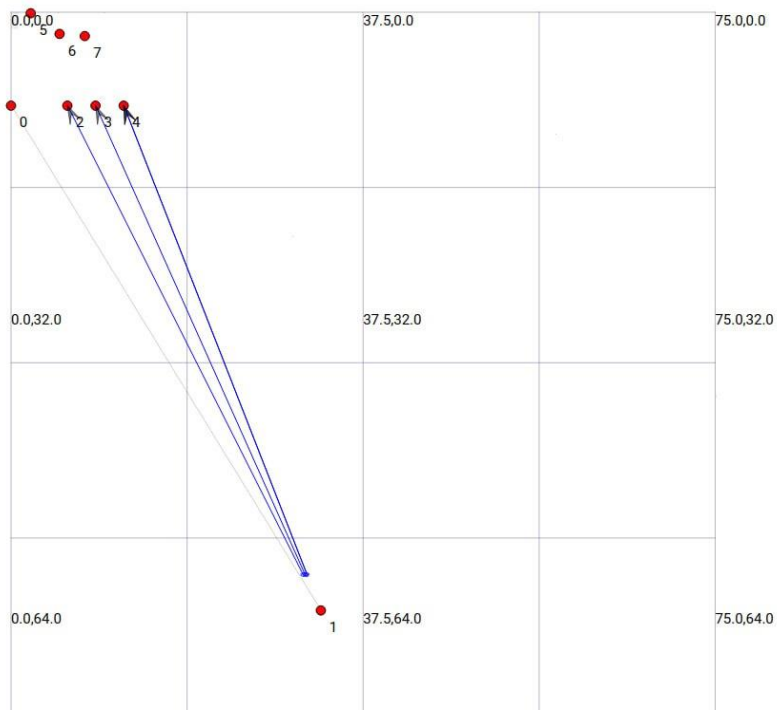


## Third.cc

```

bhuvi@bhuvi-VivoBook-ASUSLaptop-M1603QA-M1603QA:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/third.cc
[0/2] Re-checking globbed directories...
ninja: no work to do.
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01624s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01624s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.02849s client received 1024 bytes from 10.1.2.4 port 9

```



Fourth.cc

```
$ ./ns3 run fourth
```

```
Traced 0 to 1234
```

Fifth.cc

```
$ ./ns3 run fifth
```

```
1.00419 536
```

```
1.0093 1072
```

```
1.01528 1608
```

```
1.02167 2144
```

```
...
```

```
1.11319 8040
```

```
1.12151 8576
```

```
1.12983 9112
```

```
RxDrop at 1.13696
```

```
...
```

## CHAPTER 9

### CONCLUSION

In this project, we embarked on a journey to augment the ns-3 simulator with advanced capabilities tailored for 5G network simulation and analysis. Our endeavor focused on integrating Wireshark tracing, ASCII tracing, NetAnim visualization, and TCP congestion control analysis into the ns-3 framework.

Through meticulous design and implementation, we successfully infused Wireshark tracing functionality into ns-3, empowering researchers to delve into packet-level details and unravel the intricacies of 5G network communication. This feature serves as a cornerstone for precise diagnosis, protocol validation, and performance evaluation in simulated 5G environments.

Additionally, our efforts extended to incorporating ASCII tracing, facilitating the generation of comprehensive trace files capturing the dynamics of 5G network elements. These trace files serve as invaluable assets for studying mobility patterns, analyzing routing protocols, and refining network simulations to mirror real-world scenarios more accurately.

Furthermore, the integration of NetAnim visualization breathed life into our simulations, offering an intuitive platform to visualize 5G network behavior dynamically. With support for depicting UE mobility, handover events, and data flow trajectories, NetAnim enriches our understanding of network dynamics and fosters effective communication of simulation results.

Moreover, our focus on TCP congestion control analysis brought forth critical insights into the behavior of congestion control algorithms in 5G networks. By meticulously tracing congestion window fluctuations and collecting TCP performance metrics, we provided researchers with indispensable tools to evaluate and optimize congestion control mechanisms in diverse 5G deployment scenarios.

In conclusion, our project represents a significant stride forward in the realm of 5G network simulation and analysis. By equipping the ns-3 simulator with advanced functionalities tailored for 5G research, we have empowered researchers and network engineers worldwide to explore, innovate, and shape the future of wireless communication systems. As we conclude this chapter, we remain steadfast in our commitment to advancing the frontiers of 5G technology and fostering a future where seamless connectivity enriches lives across the globe.

## **BIBLIOGRAPHY**

- [1] [https://ieeexplore.ieee.org/abstract/document/9926175?casa\\_token=UH6iYO0qfrsAAAAA:W1TNAnVuhiDW7obXZQU65X\\_6b17T\\_7ejI7moWHPCFGVo9FZm7pYXQNjs3cP8uD1dIhnToUjQv8CzSg](https://ieeexplore.ieee.org/abstract/document/9926175?casa_token=UH6iYO0qfrsAAAAA:W1TNAnVuhiDW7obXZQU65X_6b17T_7ejI7moWHPCFGVo9FZm7pYXQNjs3cP8uD1dIhnToUjQv8CzSg)
- [2] <https://www.sciencedirect.com/science/article/pii/S187705092200477X> [5G NR Configured Grant in ns-3 Network Simulator for Ultra-Reliable Low Latency Communications]
- [3] [https://www.researchgate.net/publication/279459069\\_5G\\_mmWave\\_Module\\_for\\_ns-3\\_Network\\_Simulator](https://www.researchgate.net/publication/279459069_5G_mmWave_Module_for_ns-3_Network_Simulator)
- [4] [https://ieeexplore.ieee.org/abstract/document/5734730?casa\\_token=CiTV\\_dX6iJcAAAAA:5Y4wewmFzfuUISZQMe4G6Gfa5uldleV8n4PEqXPdFFDX7jykgbCJShwSjIf0GB0nK3blh04ykNP5fA](https://ieeexplore.ieee.org/abstract/document/5734730?casa_token=CiTV_dX6iJcAAAAA:5Y4wewmFzfuUISZQMe4G6Gfa5uldleV8n4PEqXPdFFDX7jykgbCJShwSjIf0GB0nK3blh04ykNP5fA) [importance of the simulation against encryption]
- [5] [https://ieeexplore.ieee.org/abstract/document/10457056?casa\\_token=QafFjSdFgroAAAAA:iCEQ19GT5YzbezFQ00s43bEVnKSIA0uRkSBGq\\_UmS1yjYvQ0QBMRPt2aKzf6\\_a6V9DB7KH9xjWsC7w](https://ieeexplore.ieee.org/abstract/document/10457056?casa_token=QafFjSdFgroAAAAA:iCEQ19GT5YzbezFQ00s43bEVnKSIA0uRkSBGq_UmS1yjYvQ0QBMRPt2aKzf6_a6V9DB7KH9xjWsC7w) [SPECTRAL TEMPORAL GRAPH NN FOR CSI PREDICTION]
- [6] [https://ieeexplore.ieee.org/abstract/document/8013810?casa\\_token=pgIosZu2h5kAAAAA:orccJH9hBXXsajmMvbQwG1fXeFqTiTjrA\\_J4twA4muYogpwyr4p01cibrDoB\\_u\\_49OzbRf0sXrKVIA](https://ieeexplore.ieee.org/abstract/document/8013810?casa_token=pgIosZu2h5kAAAAA:orccJH9hBXXsajmMvbQwG1fXeFqTiTjrA_J4twA4muYogpwyr4p01cibrDoB_u_49OzbRf0sXrKVIA) [CONTINUOUS SPECTRAL QoS TRANSFER FOR VIDEO STREAMING]
- [7] <https://link.springer.com/article/10.1186/s13638-022-02133-3> [UPLINK COVERAGE ENHANCEMENTS FOR CSI PREDICTION]
- [8] <https://www.sciencedirect.com/science/article/abs/pii/S1570870520307034> [PDCH FOR LTE/ 4G MASSIVE MIMO CSI]
- [9] <https://ieeexplore.ieee.org/abstract/document/9914925> [BSNL RAN / BRAN MULTI SPLIT O-RAN FUNCTIONALITY]
- [10] [https://web.archive.org/web/20220526051501id\\_/https://ieeexplore.ieee.org/ielx7/9668311/9667790/09668547.pdf](https://web.archive.org/web/20220526051501id_/https://ieeexplore.ieee.org/ielx7/9668311/9667790/09668547.pdf) [ENCRYPTION AND DECRYPTION IN SERVING AND PACKET GATEWAY RESPECTIVELY]