[Data Communications and Networks]

[(Open Ended Lab) Report]

[INTRODUCTION to CALLBACKS in NS-3]

Submitted by:

[Mudassar Hussain] [23L-6006]

Submitted to:
[Ma’am Abeer Bashir]

[13/10/2025]

Department of Electrical Engineering
National University of Computer and Emerging Sciences, Lahore

Table of Contents

Introduction

Introduction:

The **main idea** of this project is to explore and implement **callbacks for custom data tracing in NS-3 (Network Simulator 3)**. Network simulation is an essential part of computer networking research, enabling the study of system performance under various configurations without the need for physical hardware.

NS-3 provides tracing through **pcap** files and **logging systems**, which capture standard network events for post-simulation analysis. However, these built-in methods are sometimes restrictive, as they do not allow custom or event specific data collection. To overcome this limitation, NS-3 provides **callback-based tracing**, a more dynamic and advance approach where user-defined functions are triggered automatically when certain simulation events occur.

In this project, the callback concept is applied to monitor two main parameters:

1. **Packets received correctly**, and
2. **Packets received with errors.**

By using callbacks connected to specific trace sources (such as PhyRxEnd and PhyRxDrop), the simulation can gather real-time packet statistics. Furthermore, these callbacks are extended to collect **per-node statistics**, allowing detailed analysis of each node's network performance rather than relying on results.

The **significance** of this approach lies in its **flexibility**. Callback-based tracing allows researchers to design customized data collection systems, monitor network behavior dynamically, and respond to specific conditions during runtime.

Problem Analysis

The **problem presented** in this project focuses on enhancing the **data tracing mechanism** in NS-3 by implementing **callbacks** for custom event handling. While NS-3 provides built-in tracing tools such as pcap and logging, these tools are often limited to predefined data and do not offer flexibility for real-time or event-specific analysis. Therefore, the task is to design a mechanism that can dynamically capture specific events particularly **packet receptions and packet errors** using the callback system.

The core challenge is to track not only the total number of packets received correctly and those received with errors, but also to extend this tracing to each individual node. This per-node tracking enables a more detailed understanding of network behavior.

To **solve this problem**, callbacks are implemented and connected to **trace sources** provided by NS-3 NetDevice class, specifically PhyRxEnd (for successful receptions) and PhyRxDrop (for packet errors). Each time one of these events occurs, the corresponding callback function is triggered automatically. These functions increment counters and display results, providing real time insights into packet flow and network reliability.

Furthermore, the call back based approach is scalable additional callbacks can be added to monitor other metrics such as transmission delay, queue size, or packet loss rate. The method combines event driven programming with simulation-based data analysis, offering a flexible and precise way to collect network performance data that can later be used for visualization, optimization, or fault analysis.

Design Requirements

- **Objective:**

  To design and implement a callback-based tracing system in NS-3 that records the number of packets received correctly and those received with errors for each node.

- **Inputs:**
  - Network topology consisting of **two nodes** connected through a **Point-to-Point (P2P)** link.
  - Configuration parameters:
    - Data rate: **5 Mbps**
    - Channel delay: **2 ms**
    - Packet size: **1024 bytes**
    - Simulation duration: **10 seconds**
- **Process:**
  - Establish a client-server setup using **UDP Echo Client** and **UDP Echo Server** applications.
  - Attach callback functions to trace sources:
    - PhyRxEnd → triggers when a packet is successfully received.
    - PhyRxDrop → triggers when a packet is lost or received with errors.
  - Maintain counters to track total successful and erroneous packets per node.

- **Outputs:**
  - Display the total number of packets **received successfully**.
  - Display the total number of packets **received with errors**.
  - Provide node wise statistics for detailed performance analysis.

- **Constraints:**
  - The simulation must be lightweight and time-efficient.
  - Callback functions should not introduce significant processing overhead.
  - Accuracy in event capturing is essential for valid results.

Possible Solutions

1. Using pcap Tracing:

- Pcap tracing allows capturing packet level data during the simulation for offline analysis.
- Advantages: Generates detailed logs of packets transmitted and received.
- Limitations:
  - Data is stored in external files, requiring post processing.
  - Does not support real time data handling or event driven reactions.

2. Using NS-3 Logging System:

- The built-in logging mechanism (NS_LOG) can record events such as transmissions and receptions.
- Advantages: Easy to implement and provides textual information during runtime.
- Limitations:
  - Limited to predefined events.
  - Cannot easily gather node specific or custom performance statistics.

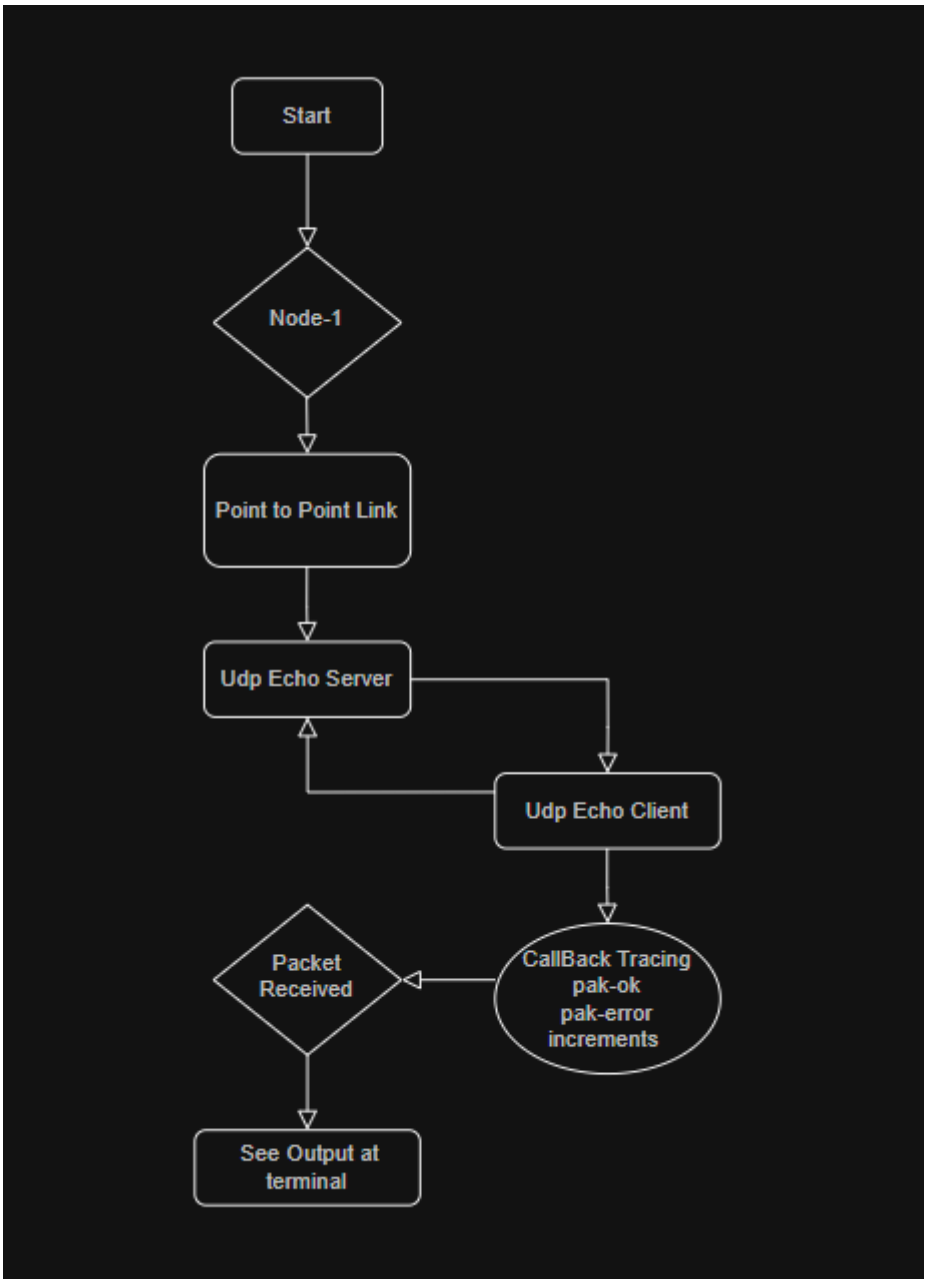
3. Using Callback-Based Tracing (Chosen Solution):

- Callbacks are user-defined functions linked directly to NS-3 trace sources.
- Advantages:
  - Event-driven and real-time data collection.
  - Allows tracking of specific events such as packet success and error per node.
  - Enables customized data handling, logging, or triggering of other actions.
  - Requires no post-processing since results are collected during simulation.
- Limitations: Slightly higher complexity in implementation compared to logging.

- **Comparison and Justification:**

  - While pcap and logging provide static tracing, callbacks offer dynamic, flexible, and interactive tracing.
  - The callback-based approach supports real-time analysis, custom event detection, and node-level statistics, making it the most suitable and efficient solution for this project.

Preliminary Design



Design Description

Design Description

1. Node Setup Block:
Two nodes are created in the simulation using the NodeContainer class.

- **Node 0** functions as the **UDP client**.
- **Node 1** functions as the **UDP echo server**.
  These nodes act as the basic communication units for data transmission and reception.

2. Communication Link Block:
A **Point-to-Point (P2P)** link is established between the two nodes using PointToPointHelper.

- Data rate: **5 Mbps**
- Propagation delay: **2 ms**
  This block defines the communication channel through which packets are exchanged.

3. Internet Stack Installation:
The **Internet stack** is installed on both nodes using InternetStackHelper.
This enables the nodes to use IP-based communication, assigning IP addresses to each device through the Ipv4AddressHelper class.

4. Application Layer Block:

- **UDP Echo Server:** Installed on Node 1 and listens on port 9.
- **UDP Echo Client:** Installed on Node 0 to send packets to the server at defined intervals.
- Packet size: **1024 bytes**, interval: **1 second**, total packets: **5.**

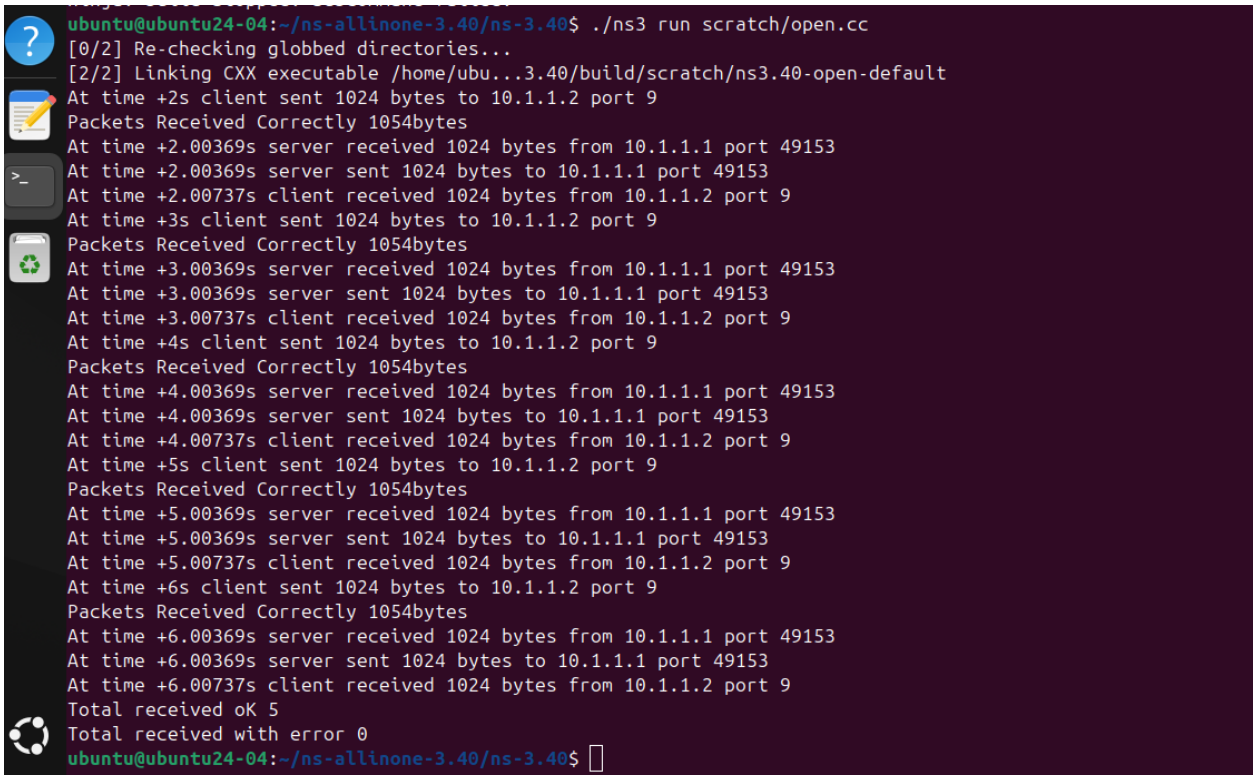
5. Callback Tracing Block:
Custom callback functions are defined and connected to specific NS-3 **trace sources**:

- PhyRxEnd → Triggers the pak_Ok callback when a packet is received correctly.
- PhyRxDrop → Triggers the pak_Error callback when a packet is dropped or received with errors.
  These callbacks increment counters to track packet statistics.

6. Statistics and Output:
At the end of the simulation, the total number of **successfully received** and **error packets** is displayed.
This provides real-time feedback on the network performance and verifies callback functionality.

7. Simulation Control:
The simulation is executed using Simulator::Run() and terminated using Simulator::Destroy().
The total execution time is set to **10 seconds**.

Software Simulation (Optional)



Experimental Results

Experimental Results

- The simulation was executed successfully using NS-3, where a **UDP Echo Client** transmitted data packets to a **UDP Echo Server** over a **Point-to-Point link**.
- The callback functions pak_Ok and pak_Error were connected to trace sources PhyRxEnd and PhyRxDrop, respectively.
- During execution, the terminal displayed real-time tracing information each time a packet was received successfully.
- The output confirmed that all packets were transmitted and received correctly without any losses.

Observed Output:

- Total packets sent: **5**
- Total packets received correctly: **5**
- Total packets received with error: **0**

Performance Analysis

Performance Analysis

- The simulation results show that all **five packets** sent by the client were **successfully received** by the server, with **zero packet errors**.

- This confirms that the implemented callback functions (pak_Ok and pak_Error) operated correctly, responding to each packet reception event in real time.

Performance Discussion:

- The **callback-based tracing system** efficiently captured and displayed packet reception statistics during simulation without the need for external log files.

- The design provided **real time monitoring**, allowing immediate feedback on packet delivery status.

- The **processing overhead** introduced by callbacks was minimal, demonstrating that this method is both **lightweight and effective** for performance monitoring.

- By extending the callbacks to gather **node specific statistics**, the solution can be scaled for larger topologies, enabling per node performance tracking.

Conclusion

Conclusion

The project successfully demonstrated the use of **callbacks for custom data tracing in NS-3**. By implementing event driven tracing functions, it was possible to monitor packet transmission and reception dynamically during the simulation. The results showed that all packets were received correctly, validating both the network setup and the callback functionality.

The callback-based approach provided significant **flexibility** and **control** over data collection compared to traditional tracing methods like pcap and logging. It enabled **real time performance monitoring** and simplified the process of gathering node specific statistics without post simulation analysis.

Through this experiment, it became evident that callbacks are a powerful feature in NS-3, allowing developers to build intelligent, responsive, and highly customizable tracing systems. The concept can be further extended to measure other network parameters or even trigger adaptive mechanisms based on live data.

In conclusion, the implementation of callbacks enhanced the **efficiency**, **accuracy**, and **analytical depth** of network simulations, making it a valuable method for both research and performance evaluation in communication networks.

References

References

1. Riley, G. F., & Henderson, T. R. (2010). *The ns-3 Network Simulator.* In *Modeling and Tools for Network Simulation.*
2. NS-3 Tutorial, *"Tracing and Logging in NS-3."* https://www.nsnam.org/docs/tutorial/html/tracing.html
3. Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer Networks* (5th ed.). Pearson Education.
4. Kurose, J. F., & Ross, K. W. (2021). *Computer Networking: A Top-Down Approach* (8th ed.). Pearson.
5. YouTube.