TEST 9: PHYSICAL LAYER - HARQ OPERATION

REFERENCE STANDARD: 3GPP TS 38.321, TS 38.214

LAYER: PHYSICAL LAYER

DIFFICULTY LEVEL: INTERMEDIATE

SUBMITTED BY:

GROUP 3 - TEAM MEMBERS:

- ***55984_Vikas Srivastava***
- ***58622_Harshinie M***
- ***58623_Shreyash Bhatt***
- ***58624_Jayavarshini G***
- ***58635_Yuvaraj P***

TABLE OF CONTENTS

1. INTRODUCTION TO HARQ

Hybrid Automatic Repeat Request (HARQ) is a core reliability mechanism in the 5G New Radio (NR) system that operates across the Physical Layer (Layer 1) and the Medium Access Control (MAC) Layer (Layer 2). It is designed to ensure reliable data delivery over the inherently error-prone wireless channel while maintaining the stringent requirements of high throughput, low latency, and spectral efficiency defined for 5G systems.

According to 3GPP specifications, HARQ combines Forward Error Correction (FEC) with retransmission control, allowing the receiver to request additional redundancy when decoding fails. Unlike classical ARQ, where erroneous packets are discarded, HARQ enables the receiver to store soft information and progressively improve decoding performance using incremental redundancy and soft combining.

3GPP References

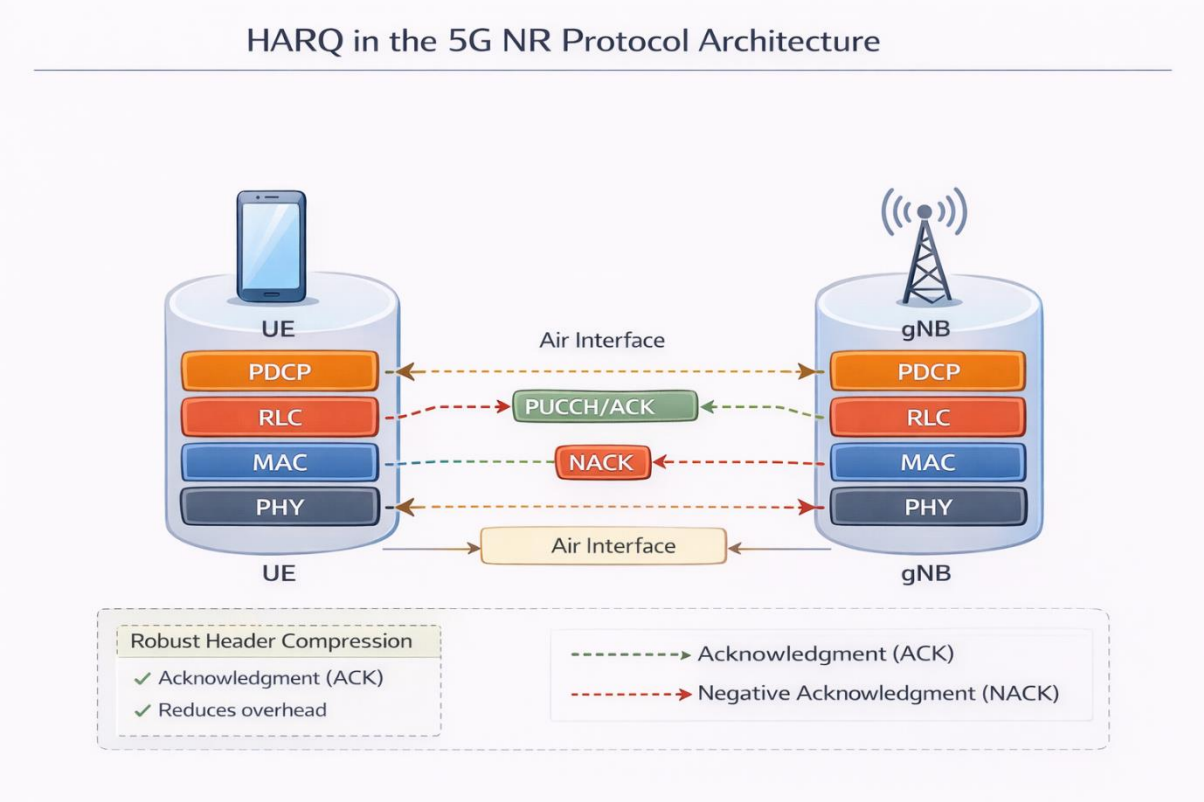
The HARQ operation implemented in this project is based on the specifications defined by the 3rd Generation Partnership Project (3GPP). The main references include 3GPP TS 38.321, which describes MAC layer procedures including HARQ operations, and 3GPP TS 38.214, which defines physical layer data transmission and retransmission mechanisms. These standards ensure alignment with HARQ behavior in 5G NR systems.

1.2 HARQ in the 5G NR Protocol Architecture

In 5G NR, HARQ is tightly integrated between the MAC layer procedures (3GPP TS 38.321) and the Physical Layer transmission mechanisms (3GPP TS 38.214). The MAC layer is responsible for managing HARQ processes, including process identification, retransmission control, and feedback timing, while the Physical Layer

handles coding, modulation, redundancy version selection, and soft combining.

This separation of responsibilities allows HARQ to operate efficiently across multiple parallel processes. Each HARQ process independently manages a Transport Block (TB), enabling continuous data flow and minimizing idle time on the radio interface. This architecture is essential for supporting 5G use cases such as eMBB (enhanced Mobile Broadband), URLLC (Ultra-Reliable Low Latency Communications), and mMTC (massive Machine-Type Communications).



1.3 Motivation for HARQ Simulation

Understanding HARQ behavior is critical for anyone working with 5G PHY and MAC layer design, testing, or optimization. While 3GPP specifications define the procedures in detail, practical insight is

gained by observing how HARQ processes evolve over time, how retransmissions are triggered, and how decoding performance improves with each redundancy version.

This project focuses on simulating the HARQ operation at the Physical Layer level, including:

- HARQ process management
- New Data Indicator (NDI) handling
- Redundancy Version (RV) cycling
- ACK/NACK feedback processing
- Soft combining and retransmission control

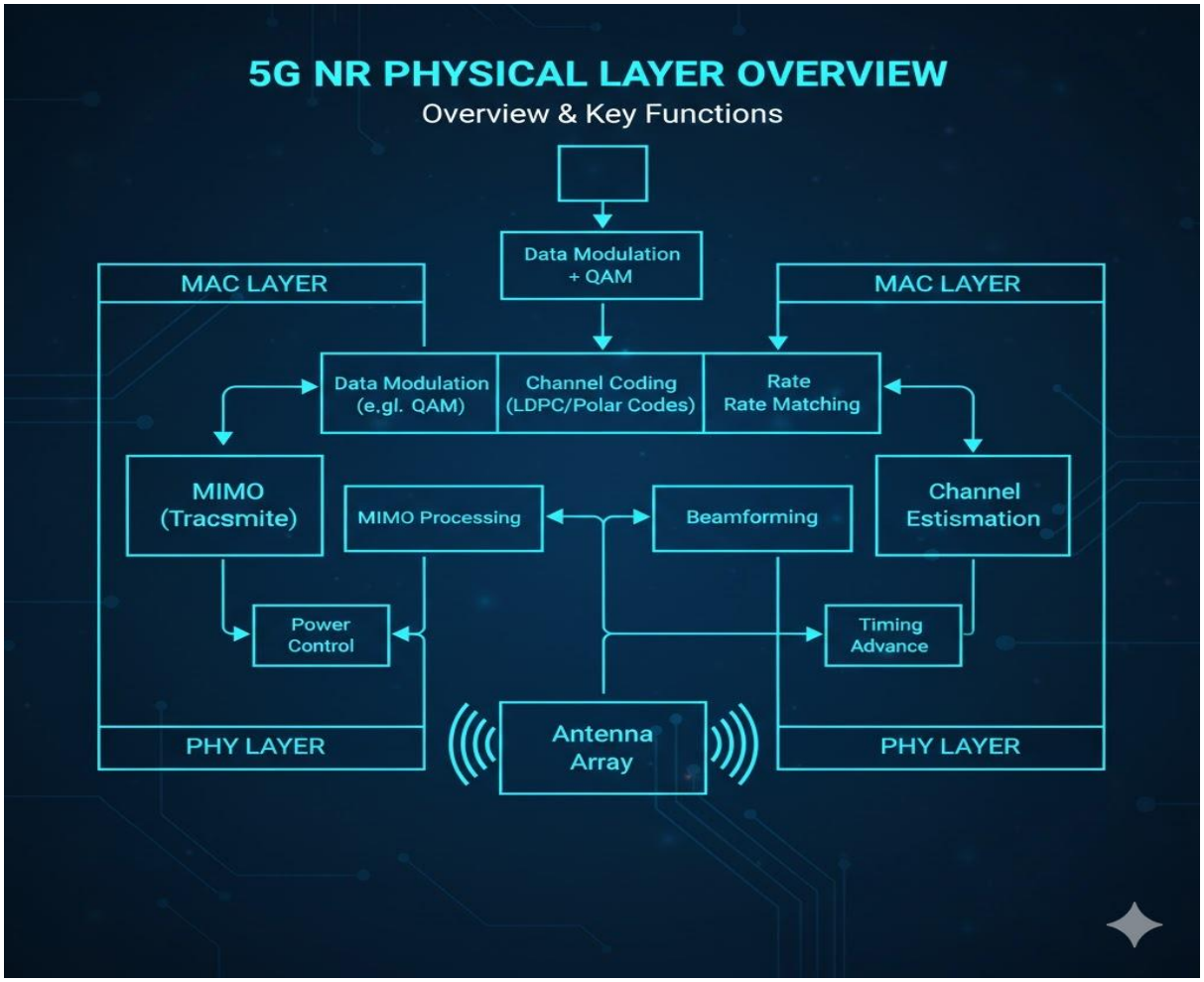
By implementing these mechanisms in C++, the simulator provides a controlled environment to study how HARQ improves reliability and efficiency in 5G NR systems.

2. 5G NR PHYSICAL LAYER OVERVIEW

The Physical Layer (Layer 1) in 5G NR is responsible for the actual transmission and reception of data over the radio interface. It converts higher-layer data into radio signals for transmission and performs the reverse operation at the receiver. This layer is designed to support the key 5G requirements of high data rates, low latency, and flexible deployment across diverse spectrum bands.

At a functional level, the Physical Layer handles:

- Channel coding and rate matching
- Modulation and demodulation
- Resource element mapping and scheduling support
- HARQ-related transmission and reception procedures



In the 5G NR protocol stack, the Physical Layer works closely with the MAC layer. While the MAC layer manages HARQ process control, scheduling, and feedback timing (as defined in TS 38.321), the Physical Layer executes the actual data transmission procedures, including selecting the Redundancy Version (RV), performing soft combining, and delivering ACK/NACK feedback (as specified in TS 38.214).

The Physical Layer also supports flexible numerology, scalable bandwidths, and multiple transmission schemes (e.g., MIMO, beamforming), enabling 5G NR to adapt to different deployment scenarios such as sub-6 GHz and mmWave bands.

3. HARQ ARCHITECTURE AND PROCESS MANAGEMENT

3.1 HARQ Architecture in 5G NR

In 5G NR, HARQ operates through multiple parallel HARQ processes, enabling continuous data transmission without waiting for feedback from previous transmissions. This parallelism ensures high throughput and minimizes idle radio resources.

Each HARQ process is uniquely identified by a HARQ Process ID, which is signaled in Downlink Control Information (DCI). The use of asynchronous HARQ in NR allows retransmissions to be scheduled flexibly, unlike synchronous HARQ used in earlier technologies.

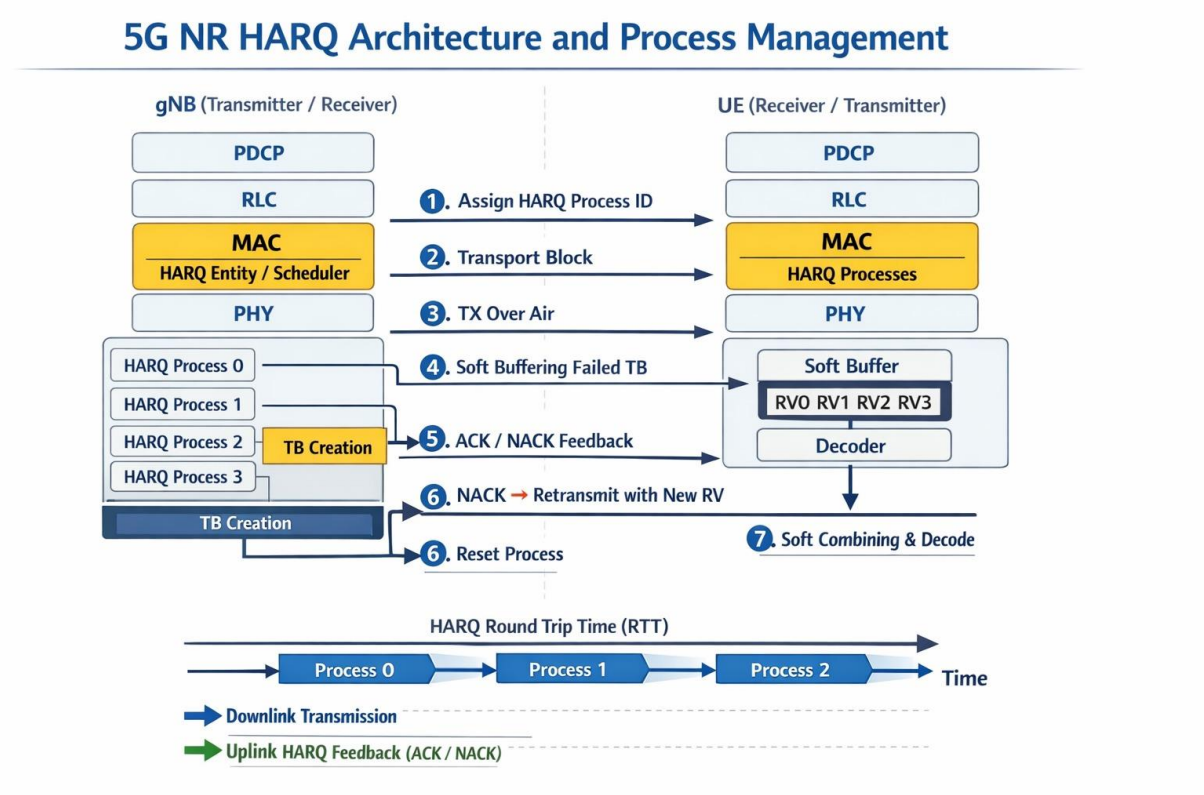
The HARQ architecture consists of two tightly coordinated parts:

- The MAC layer is responsible for HARQ process allocation, tracking retransmission counters, managing ACK/NACK feedback, and scheduling retransmissions.
- The Physical Layer performs channel coding, rate matching, redundancy version selection, transmission of transport blocks, and soft combining at the receiver.

Each HARQ process maintains its own state, buffers, and retransmission parameters, allowing independent handling of multiple transport blocks.

3.2 HARQ Process Management

HARQ process management ensures proper coordination between transmission attempts, feedback reception, and retransmission control. The following table summarizes the lifecycle and management behavior of a HARQ process.



3.3 Key Parameters in HARQ Process Management

Parameter	Description
HARQ Process ID	Identifies one of multiple parallel HARQ processes
NDI (New Data Indicator)	Indicates new TB or retransmission
RV (Redundancy Version)	Determines which parity bits are transmitted
Retransmission Counter	Tracks number of retransmissions
Soft Buffer	Stores LLRs for soft combining
Maximum Retransmissions	Configured limit before declaring failure

3.4 Parallel HARQ Operation in NR

5G NR supports multiple HARQ processes running in parallel so that new data can be transmitted while previous transmissions are still awaiting feedback. Each process operates independently with its own buffer and state, allowing continuous use of radio resources without idle gaps.

By overlapping transmission and feedback across several HARQ processes, the system maintains high throughput and low latency even under changing channel conditions. Figure X illustrates how multiple HARQ processes operate simultaneously to keep the transmission pipeline full.

4. HARQ OPERATION: NDI, RV CYCLING, ACK/NACK, SOFT COMBINING & RETRANSMISSION

HARQ in 5G NR is not just a retransmission mechanism—it is a closed-loop reliability system that continuously adapts to channel conditions. By combining forward error correction with feedback-driven retransmissions, HARQ ensures that data delivery remains robust without sacrificing spectral efficiency or latency.

Each HARQ process behaves like a self-contained transmission pipeline. It controls when new data is injected, how redundancy is increased over time, and when a Transport Block is finally accepted or discarded. This section describes the internal logic that governs this behavior.

4.1 NDI: Controlling the Identity of Data

The New Data Indicator (NDI) acts as the identity switch for a HARQ process. It distinguishes a brand-new Transport Block from a retransmission of an earlier one.

When the transmitter toggles NDI, it signals the start of a new data lifecycle. The receiver immediately resets its combining buffer and prepares to decode fresh information. If NDI remains unchanged, the receiver interprets the transmission as part of an ongoing HARQ attempt and performs soft combining with previously stored data.

In essence, NDI ensures that both ends of the link stay aligned on what data is being processed.

4.2 RV Cycling: Progressive Redundancy Injection

The Redundancy Version (RV) controls which part of the coded data is transmitted in each HARQ attempt. Rather than repeating the same bits, the system progressively injects new parity information with every retransmission.

The standard RV progression:

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3$$

represents a carefully designed redundancy schedule. Each step adds new decoding power at the receiver, increasing the probability of success without wasting radio resources on blind repetition.

RV cycling turns retransmission into an intelligent refinement process, not just a retry.

4.3 ACK/NACK: The Closed-Loop Control Signal

ACK/NACK feedback is the decision point in the HARQ loop. After decoding a Transport Block, the receiver reports either:

- ACK — “I have decoded this block successfully.”
- NACK — “I need more redundancy to decode this block.”

An ACK immediately terminates the HARQ process and frees the associated buffers. A NACK triggers a controlled escalation: the same Transport Block is retransmitted using the next RV value, keeping the process identity intact.

This feedback loop allows the transmitter to adapt in real time to channel quality without over-transmitting.

4.4 Soft Combining: Accumulating Reliability

Soft combining is where HARQ becomes truly powerful. Instead of discarding failed transmissions, the receiver stores probability-level information from each attempt and merges it with new information from retransmissions.

With each combining step, the receiver's confidence in the decoded bits increases. This mechanism transforms noisy, unreliable receptions into a coherent and decodable signal over time.

Incremental redundancy ensures that every retransmission contributes new knowledge, making HARQ both reliable and efficient.

4.5 Retransmission Logic and Failure Resolution

Every HARQ process enforces a retransmission limit. With each NACK, a counter is incremented and the next RV is scheduled. This continues until either:

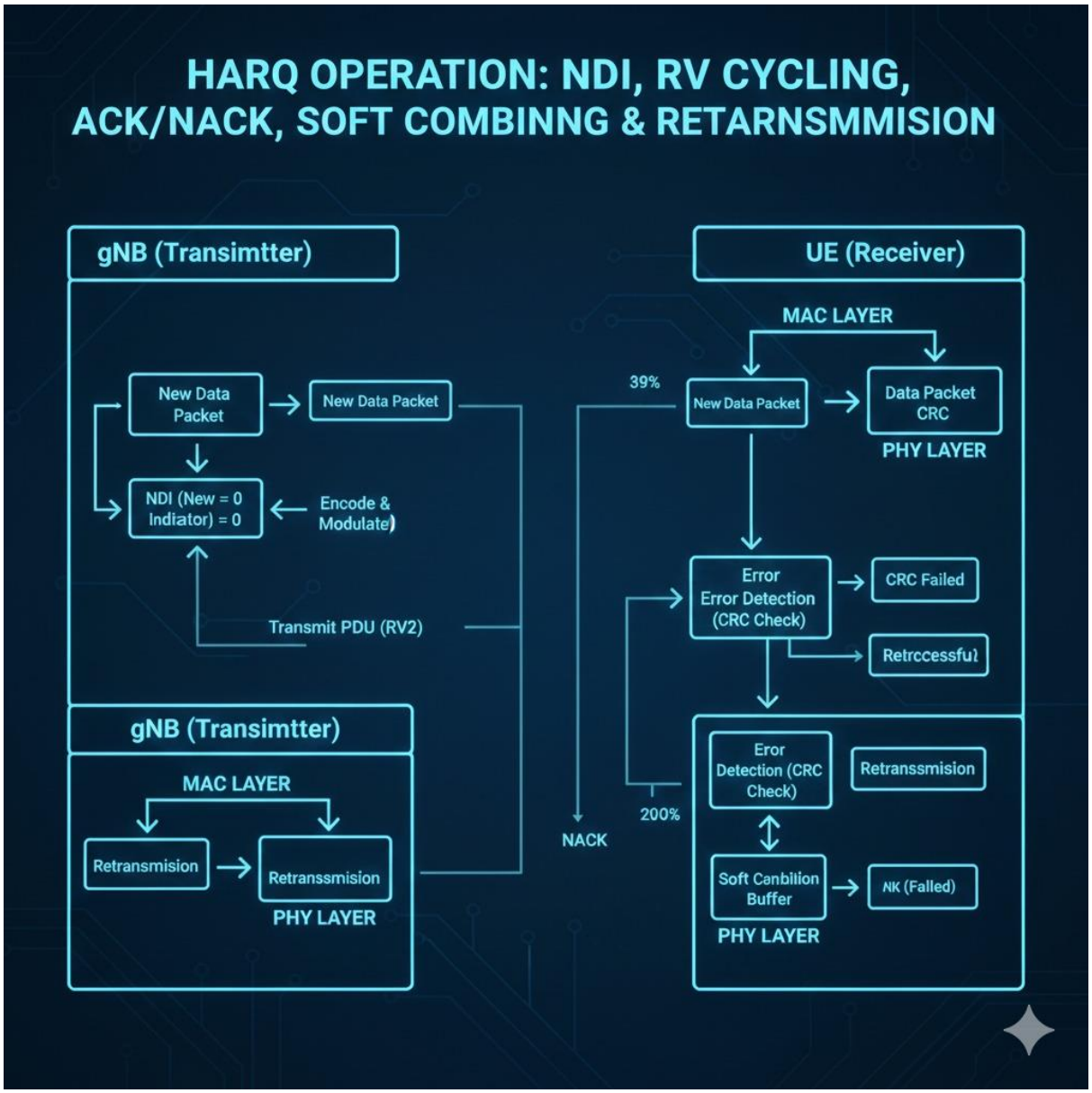
- The Transport Block is successfully decoded (ACK), or
- The maximum retransmission threshold is reached

If the threshold is exceeded, the block is declared failed and the process is reset. This prevents infinite retries and preserves system stability.

4.6 Operational Flow Summary

1. A HARQ process is assigned to a new Transport Block.
2. NDI toggles → new data transmission with RV = 0.
3. Receiver decodes and returns ACK/NACK.
4. On NACK → RV is incremented and retransmission is scheduled.
5. Receiver performs soft combining.

6. The cycle repeats until success or controlled failure.



5. C++ IMPLEMENTATION

This C++ program simulates the 5G NR HARQ (Hybrid Automatic Repeat reQuest) process based on 3GPP TS 38.321 5.4. It models a HARQ process structure containing parameters such as NDI (New Data Indicator), redundancy version (RV), retransmission count, and process state. The code demonstrates new transmission,

retransmission after NACK, and successful acknowledgment (ACK) handling. The RV sequence (0 → 2 → 3 → 1) represents incremental redundancy used during retransmissions. A random generator is included to emulate channel behavior, while console outputs show HARQ operation step-by-step. Overall, the program explains stop-and-wait HARQ behavior and retransmission control in a simplified simulation form.

5.1 Step 1 : HARQ Process Structure and Transmission Logic

This part of the C++ code defines the HARQ process structure used to simulate the Hybrid Automatic Repeat request mechanism in 5G NR. It models the internal parameters required for reliable data transmission, including the New Data Indicator (NDI), redundancy version (RV), retransmission count, and process activity status. The structure represents how a HARQ entity manages new transmissions, handles retransmissions after failures, and stops the process after reaching the maximum retry limit. The implementation follows the incremental redundancy concept defined in 3GPP standards, where different redundancy versions are used during retransmissions to improve decoding success at the receiver.

Key Components in this Code :

- HARQ Process ID – Identifies each HARQ process operating in parallel.
- NDI (New Data Indicator) – Toggles to indicate whether the transmission carries new data or retransmitted data.
- Redundancy Version (RV) – Controls the redundancy pattern used during retransmissions (0, 2, 3, 1 sequence).

- Retransmission Counter – Tracks the number of retransmission attempts.
- Maximum Retransmission Limit – Stops the process after reaching the allowed retry count.
- Process State (Active/Inactive) – Indicates whether the HARQ process is currently transmitting data.

```cpp
// HARQ Process Simulation - TS 38.321 §5.4
#include <iostream>
#include <vector>
#include <cstdint>
#include <string>
#include <array>
#include <random>

struct HARQProcess {
    int id;
    bool ndi = false; // New Data Indicator
    int rv = 0;       // Redundancy Version (0,2,3,1 cycle)
    int retx_count = 0;
    int max_retx = 4;
    bool active = false;
    std::string data;
    static const std::array<int, 4> rv_sequence;

    void new_transmission(const std::string& d) {
        data = d; ndi = !ndi; rv = 0; retx_count = 0; active = true;
        std::cout << "  HARQ[" << id << "] NEW TX: NDI=" << ndi
                  << " RV=" << rv << " data=\"" << data << "\"\n";
    }
    bool retransmit() {
        if (retx_count >= max_retx) {
            std::cout << "  HARQ[" << id << "] MAX RETX reached!\n";
            active = false; return false;
        }
        retx_count++;
        rv = rv_sequence[retx_count % 4];
        std::cout << "  HARQ[" << id << "] RETX #" << retx_count
                  << ": RV=" << rv << "\n";
        return true;
    }
    void ack() {
        std::cout << "  HARQ[" << id << "] ACK received\n";
        active = false;
    }
};
const std::array<int, 4> HARQProcess::rv_sequence = {0, 2, 3, 1};
```

5.2 Step 2 : HARQ Process Execution and Simulation Flow

The main function implements the execution flow of the HARQ simulation, demonstrating how a HARQ process behaves during transmission, retransmission, and acknowledgment stages. It initializes the HARQ process, starts new data transmission, and simulates channel conditions such as NACK (transmission failure) and ACK (successful decoding). The flow shows how retransmissions are triggered using different redundancy versions until successful reception occurs. This section provides a step-by-step demonstration of stop-and-wait HARQ behavior as defined in 3GPP TS 38.321.

Key Operations in Main Function :

- Simulation Initialization – Displays HARQ configuration details such as parallel processes and RV sequence.
- New Transmission Start – Sends initial PDU data using the HARQ process.
- NACK Simulation – Represents channel error causing retransmission requests.
- Retransmission Handling – Calls retransmission function with updated RV values.
- ACK Handling – Stops the HARQ process after successful decoding.
- Multiple Data Transmission – Demonstrates another transmission cycle to show repeated HARQ operation.

This part connects all HARQ functions together and clearly illustrates how HARQ improves reliability through controlled retransmissions and acknowledgment feedback.

```cpp
int main() {
    std::cout << "=== HARQ Process (TS 38.321 §5.4) ===\n\n";
    std::cout << "16 parallel stop-and-wait HARQ processes\n";
    std::cout << "RV sequence: 0 -> 2 -> 3 -> 1\n\n";
    std::mt19937 rng(42);
    HARQProcess proc{0};
    proc.new_transmission("PDU-DATA-A");
    // Simulate NACK (channel error)
    std::cout << "  << NACK (CRC fail) >>\n";
    proc.retransmit();
    std::cout << "  << NACK >>\n";
    proc.retransmit();
    std::cout << "  << ACK (combining success) >>\n";
    proc.ack();
    std::cout << "\n";
    proc.new_transmission("PDU-DATA-B");
    std::cout << "  << ACK (first attempt) >>\n";
    proc.ack();
    return 0;
}
```

6. SAMPLE OUTPUT

The output shows the execution of a 5G NR HARQ process where a new data transmission initially fails due to a simulated CRC error, resulting in NACK responses. Retransmissions are triggered with different redundancy versions (RV=2 and RV=3), demonstrating incremental redundancy combining until successful decoding occurs. After receiving an ACK, the HARQ process becomes inactive and completes the transmission cycle. A second data transmission is then sent and successfully acknowledged in the first attempt, showing normal HARQ operation.

```
=== HARQ Process (TS 38.321 §5.4) ===

16 parallel stop-and-wait HARQ processes
RV sequence: 0 -> 2 -> 3 -> 1

  HARQ[0] NEW TX: NDI=1 RV=0 data="PDU-DATA-A"
  << NACK (CRC fail) >>
  HARQ[0] RETX #1: RV=2
  << NACK >>
  HARQ[0] RETX #2: RV=3
  << ACK (combining success) >>
  HARQ[0] ACK received

  HARQ[0] NEW TX: NDI=0 RV=0 data="PDU-DATA-B"
  << ACK (first attempt) >>
  HARQ[0] ACK received
```

7. KEY CONCEPTS

- **Hybrid Automatic Repeat Request (HARQ)**

  A reliability mechanism that combines forward error correction with feedback-based retransmissions to improve decoding success over unreliable radio channels.
- **HARQ Process**

  An independently managed transmission pipeline identified by a unique process ID, maintaining its own buffers, redundancy state, and retransmission counter.
- **Transport Block (TB)**

  A unit of data delivered from higher layers to the Physical Layer for transmission and HARQ processing.
- **New Data Indicator (NDI)**

  A control flag used to distinguish a new Transport Block from a retransmission within the same HARQ process.
- **Redundancy Version (RV)**

An indicator that determines which portion of the coded data is transmitted during each HARQ attempt, enabling incremental redundancy.

- **ACK / NACK Feedback**

  Receiver feedback indicating successful or failed decoding, used to control HARQ retransmissions.

- **Soft Combining**

  A decoding technique where soft information from multiple transmissions is accumulated to improve decoding probability.

- **Incremental Redundancy**

  A retransmission strategy where each attempt provides new parity information instead of repeating previously sent data.

- **Maximum Retransmission Limit**

  A configured threshold that prevents indefinite retransmissions and ensures system stability.

8. COMPILATION & EXECUTION GUIDE

8.1 Prerequisites :

This simulation requires a C++11 (or later) compatible compiler. Recommended compilers include GCC 7+, Clang 5+, or MSVC 2017+. The program uses only the C++ Standard Library, so no external libraries are required.

8.2 Compilation Commands

- Windows (MinGW / Git Bash):

```
g++ code_final.cpp -o Test9_HARQ
./Test9_HARQ.exe
```

- Linux / macOS:

```
g++ code_final.cpp -o Test9_HARQ
./Test9_HARQ
```

- Windows (MSVC):

```
cl /EHsc code_final.cpp
/Fe:Test9_HARQ.exe
```

9. CONCLUSION :

This documentation presents a comprehensive study of HARQ operation in the 5G NR Physical Layer, focusing on the mechanisms that ensure reliable and efficient data transmission. By examining HARQ process management, New Data Indicator handling, Redundancy Version cycling, ACK/NACK feedback, soft combining, and retransmission control, the report highlights how reliability is achieved without compromising latency or spectral efficiency.

The accompanying C++ simulation reinforces the theoretical concepts by demonstrating HARQ behavior through observable state transitions, feedback-driven retransmissions, and performance statistics. Together, the theoretical explanation and practical implementation provide a clear understanding of HARQ as a closed-loop reliability system that plays a critical role in modern 5G communication systems.

This work serves as a strong foundation for further exploration of advanced Physical Layer features such as adaptive modulation and coding, link adaptation, and cross-layer optimization in 5G NR.