PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

Course Code: CIT-111

SUBMITTED TO:

Dr. Md Samsuzzaman

Department of Computer And Communication Engineering

Faculty of Computer Science And Engineering

SUBMITTED BY:

Name: MD Noushad Bhuiyan

ID: 2102038, Registration No: 10165

Faculty of Computer Science and Engineering

1. Introduction

Computer networks often experience:

- Packet loss
- Bit corruption
- Delay
- Duplicate packets

UDP provides fast communication but **no reliability**. To achieve correctness, reliability must be implemented manually. This project implements the **RDT 3.0 Stop-and-Wait ARQ protocol** using Python sockets.

An additional module called **unreliable_channel.py** simulates an imperfect network by introducing packet loss, corruption, and delay.

This project demonstrates the complete workflow of a reliable communication system.

2. Objectives

- Implement reliable communication over UDP
- Use sequence numbers
- Use checksum for error detection
- Implement timeout-based retransmission
- Simulate channel impairments
- Observe and analyze protocol behavior

3. System Architecture

3.1 Components

1. **Sender**
   - Prepares DATA packets
   - Sends to channel
   - Waits for ACK
   - Retransmits on timeout
2. **Receiver**
   - Validates checksum
   - Ensures correct sequence
   - Sends ACK
3. **Unreliable Channel**
   - Forwards packets
   - Randomly drops/corrupts/delays packets
4. **Common Module**
   - Packet structure
   - Checksum calculation
   - Codec for packet parse/build

4. Packet Format

| TYPE (1 byte) | SEQ (1 byte) | CHECKSUM (2 bytes) | PAYLOAD (N bytes) |

TYPE:

- 0 → DATA
- 1 → ACK

Sequence Number:

- Alternates between **0 and 1** (Stop-and-Wait)

Checksum:

- 16-bit sum of header + payload

5. Working Principle

5.1 Sender Workflow

```
Send packet → Start timer →
  If ACK received:
     send next packet
  Else (timeout):
     retransmit
```

5.2 Receiver Workflow

```
Receive packet →
  If corrupted:
    send duplicate ACK
  If correct seq:
    deliver + send ACK
  If duplicate:
    resend ACK
```

6. ASCII Diagram of Communication

```
SENDER                CHANNEL               RECEIVER


 P0 ---------------------> [drop? corrupt? delay?] ----> P0

                  <-------------------------- ACK0

 RECV ACK0

 P1 ---------------------> [.........] ----------------> P1

                  <-------------------------- ACK1
```

If any ACK or DATA is lost:

SENDER timeout → retransmit

7. Experimental Results

- **7.1 No Loss/Corruption**

drop=0, corrupt=0

Retransmissions = 0

Delivery = Perfect

- **7.2 30% Corruption**

Receiver prints:

CORRUPTED packet → sending duplicate ACK

Retransmissions = moderate

- **7.3 30% Loss**

Sender prints:

TIMEOUT → Retransmitting packet

- **7.4 Heavy Loss + Delay**

drop=0.4, corrupt=0.3, delay=0.2

Retransmissions increased significantly

8. Conclusion

This project successfully demonstrates the implementation of **Reliable Data Transfer (RDT)** using **Stop-and-Wait ARQ**.

The system:

- Detects corruption with checksum
- Handles loss with retransmission
- Manages duplicates using sequence numbers
- Simulates real-world network behaviors

The experiment clearly shows how reliability is built on top of an unreliable UDP channel.