# **Documentation: Pipelined Reliable Transfer Protocol**

## **Overview**

This project implements a connection-oriented, reliable, pipelined transport protocol similar to TCP. The protocol supports reliable data transfer between a sender and receiver using **flow control**, **congestion control**, **packet sequencing**, and **error handling**.

## **1. Protocol Design**

The protocol includes key features of TCP, such as **reliability**, **flow control**, **congestion control**, and **connection management**. We use **UDP sockets** to build these features from scratch, as UDP is a lightweight, connectionless protocol.

### **1.1 Connection Management**

* **Connection Setup**: A simple handshake is used to establish the connection, ensuring both ends are ready to communicate.
* **Connection Teardown**: A **FIN-ACK** exchange is used to close the connection.

### **1.2 Reliable Data Transfer**

Reliability is achieved using **sequence numbers**, **acknowledgments (ACKs)**, and **retransmissions**:

* **Packet Sequencing**: Each packet has a unique sequence number to track packet order and detect missing packets.
* **Acknowledgments**: Each received packet is acknowledged to confirm successful delivery.
* **Sliding Window**: A sliding window allows multiple packets to be in transit, improving throughput.

### **1.3 Flow Control**

Flow control prevents the sender from overwhelming the receiver's buffer:

* **Receiver Advertised Window**: The receiver advertises its available buffer space, and the sender adjusts the number of packets accordingly.

### **1.4 Congestion Control**

Congestion control ensures fair use of network resources:

* **Congestion Window (cwnd)**: Limits the number of unacknowledged packets in the network.
* **AIMD (Additive Increase, Multiplicative Decrease)**:
  + **Slow Start**: Starts with a small cwnd and increases it exponentially until a threshold (ssthresh) is reached.
  + **Congestion Avoidance**: After reaching ssthresh, cwnd increases linearly.
  + **Packet Loss Handling**: On packet loss, cwnd is reduced to control congestion.

## **2. Protocol Implementation**

The protocol is implemented using **UDP sockets** with the following components:

### **2.1 Core Functions**

* **Connection Setup and Teardown**:
  + connection\_setup(): Establishes the connection using a simple handshake.
  + connection\_teardown(): Closes the connection using a FIN-ACK exchange.
* **Reliable Data Transfer**:
  + rdt\_send(data): Sends data reliably by adding sequence numbers and using a timeout mechanism for retransmissions.
  + rdt\_receive(): Receives data reliably by ensuring correct packet order and sending ACKs.
* **Sliding Window Mechanism**: Maintains a window of packets that can be sent without waiting for an acknowledgment, enabling pipelining.
* **Flow Control and Congestion Control**:
  + **Flow Control**: Uses a receiver window to control the sender's data rate.
  + **Congestion Control**: Implements AIMD to adjust cwnd based on network conditions.

### **2.2 Handling Loss and Errors**

To simulate packet loss and errors:

* **Random Packet Loss**: Packet drops are introduced at random intervals to simulate network unreliability.
* **Simulation Location**: Loss is simulated during rdt\_send() to test retransmissions.

## **4. Conclusion**

This project involved designing and implementing a reliable, connection-oriented, pipelined transport protocol using **UDP** sockets. The protocol includes **flow control** and **congestion control**, similar to TCP. Testing with simulated packet loss and Wireshark analysis confirms that the protocol meets its design goals.