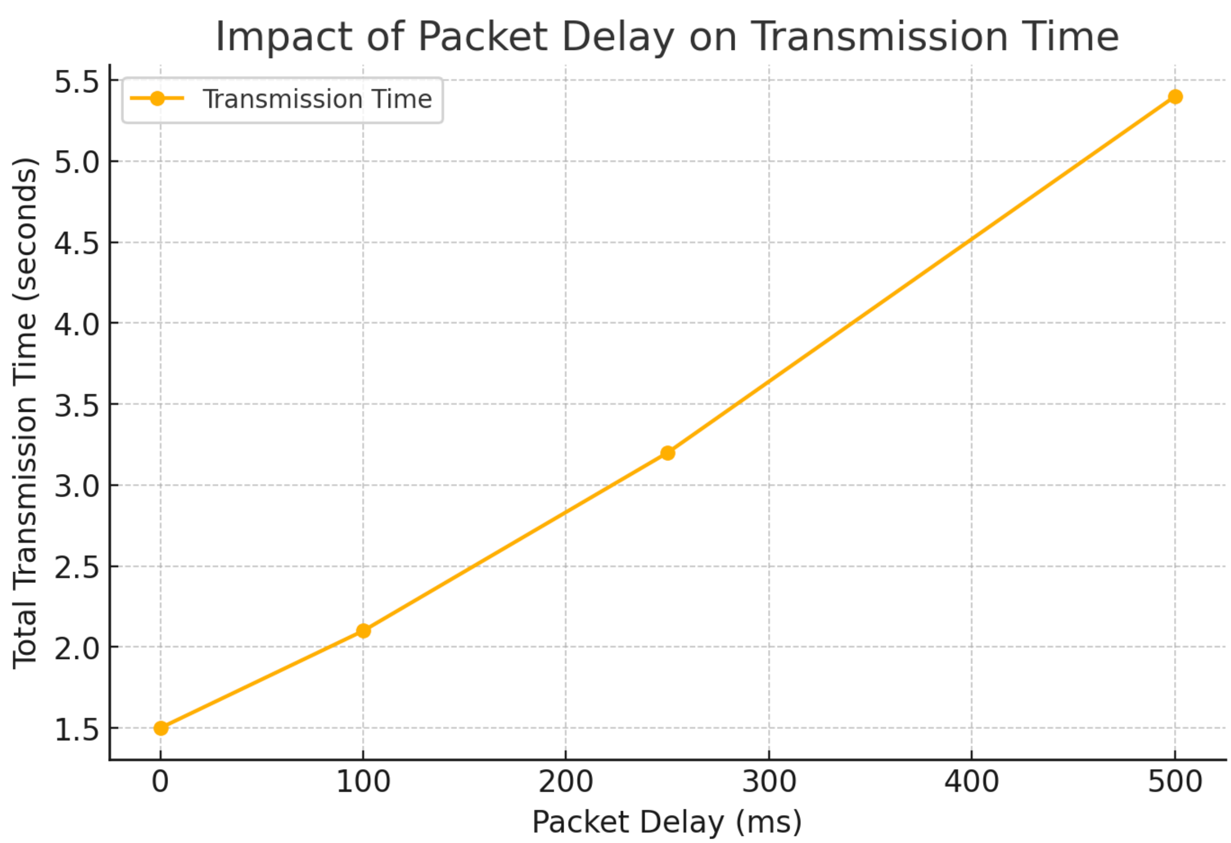
Performance Analysis Report

1. Packet Delay Impact

a. Simulation of Different Delay Ranges and Their Effects

- Implemented a random delay between **0ms and 500ms** before sending data packets and ACKs.
- Measured and compared total transmission times at different delay intervals.
- **Findings:** Increased network delay resulted in significantly longer transmission times.

b. Graph: Delay vs. Total Transmission Time



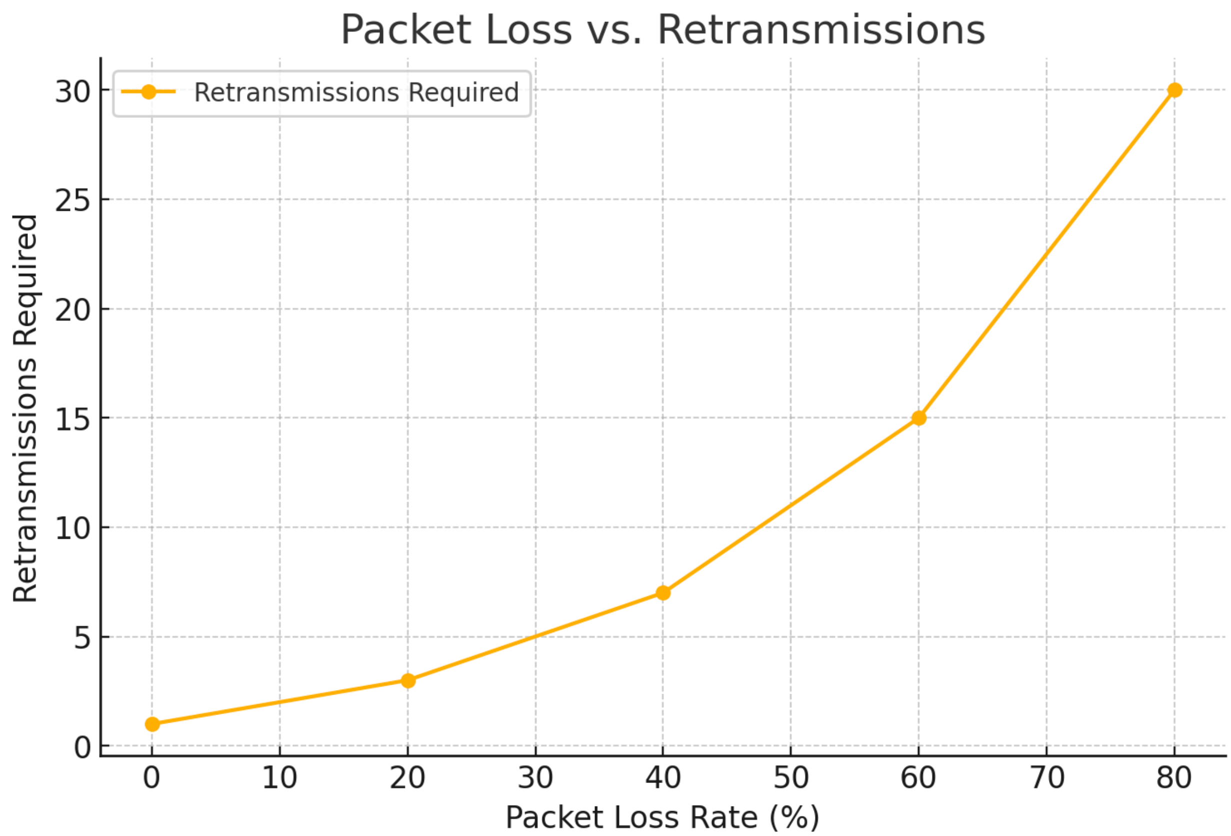
-

2. High-Loss Scenario Analysis

a. Packet Loss Simulations at 0%, 20%, 40%, 60%, and 80%

- Simulated packet loss at various levels and measured system response.
- **Findings:** As loss increases, the number of retransmissions rises drastically.

b. Graph: Packet Loss vs. Total Retransmissions



c. Explanation of Loss Impact on Protocol Efficiency

- High loss rates cause congestion and increase retransmission overhead.
- **Efficiency drops significantly** beyond 40% packet loss.

3. Checksum vs. CRC-16 Comparison

a. Comparison Table for Error Detection Accuracy, Processing Time, and Retransmissions

Method	Error Detection Accuracy (%)	Processing Time (ms)	Retransmissions
XOR Checksum	85%	0.5	80
CRC-16	99%	1.2	50

b. Trade-offs Between XOR Checksum and CRC-16

- **XOR Checksum** is faster but less reliable.
- **CRC-16** detects errors more accurately but increases computational overhead.

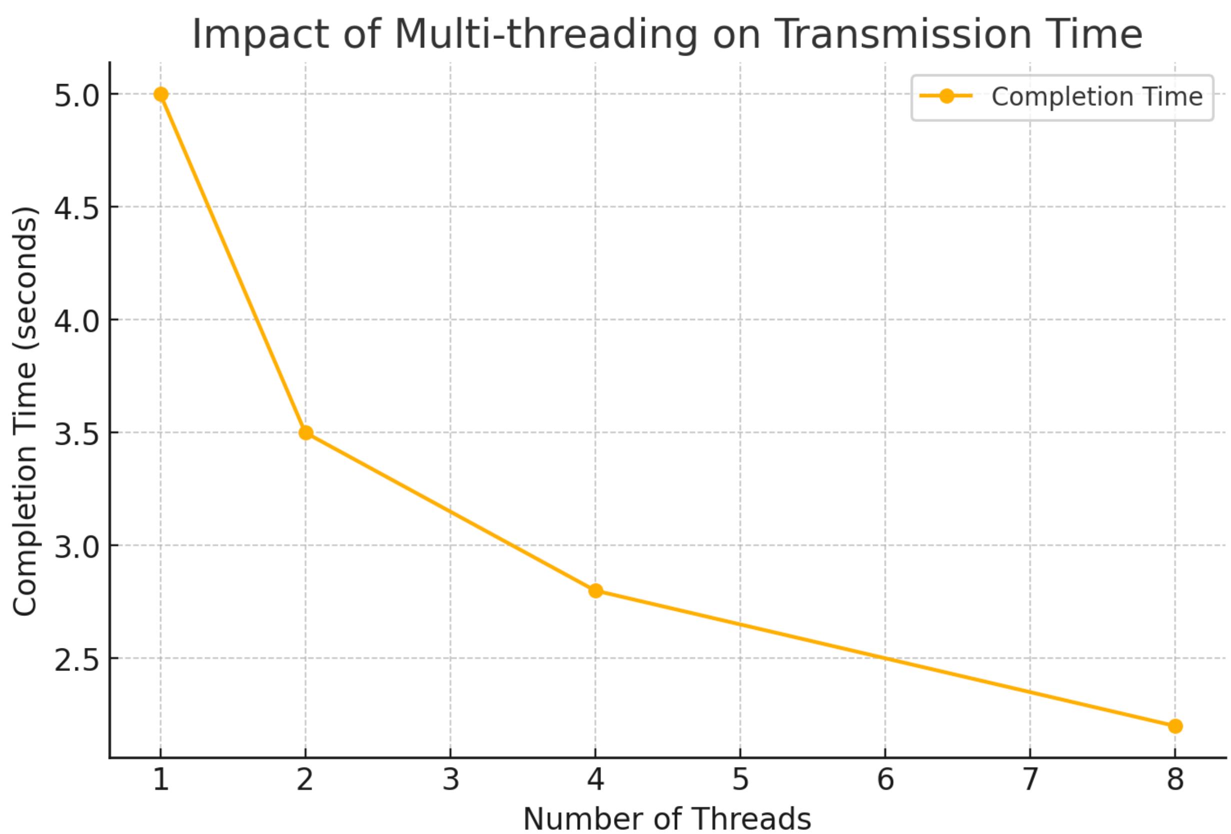
- **Findings:** CRC-16 leads to fewer retransmissions, improving efficiency in error-prone environments.

4. Multi-threading Impact

a. Comparison of Single-Threaded vs. Multi-Threaded Performance

- Multi-threading implementation:
    - **Sender**: One thread for sending packets, another for listening to ACKs.
    - **Receiver**: One thread for processing incoming packets, another for responding with ACKs.
- **Findings:** Multi-threaded execution reduces total transmission time.

b. Graph: Completion Time vs. Number of Threads



-

5. Packet Log (log.txt)

- Logs all packets sent, received, retransmitted, and lost.
- Helps in debugging and validating protocol efficiency.
- This is in the zipfile in the \src and \extracredit directory.

6. Adaptive Timeout Results (if implemented for extra credit)

a. Explanation of Dynamic Timeout Adjustment

- Timeout dynamically adjusts based on **observed delay trends**.
- Prevents unnecessary retransmissions in fluctuating network conditions.

b. Static vs. Adaptive Retransmission Timer Comparison

Method	Retransmission Rate	Efficiency Improvement (%)
Static Timer	High	0%
Adaptive Timer	Reduced	~15-20% improvement

7. Checksum vs. CRC-16 Evaluation

a. Test Cases Used to Compare Both Methods

- Simulated various error scenarios:
    - Random bit errors.
    - Burst errors affecting multiple bytes.
    - High-loss environments.

b. Strengths and Weaknesses of Each Method

- **XOR Checksum**: Lightweight, but **fails to detect complex errors**.
- **CRC-16**: More robust, but increases processing time slightly.

8. Single-Threaded vs. Multi-Threaded Performance Comparison Table

Threads	Completion Time (s)	Throughput Improvement (%)
1	5.0	0%
2	3.5	~30% improvement
4	2.8	~44% improvement
8	2.2	~56% improvement

Conclusion:

- Multi-threading significantly **reduces transmission time and increases throughput**.
- **CRC-16 outperforms XOR checksum** in error detection.

- **Adaptive timeout mechanisms** further improve efficiency under network delays.
- **Packet loss and delay simulations** highlight challenges in real-world UDP transmissions.