**Technical Report**

**UDP-based reliable data transfer algorithm.**

**Submitted By – Chinmay Patil**

**UTA id – 1002066636**

**1. Introduction**

The implemented protocol aims to facilitate efficient and reliable file transfer through a hybrid approach, combining TCP for the control channel and UDP for the data channel. This report provides a detailed explanation of the protocol's functionalities, out-of-the-box thinking, and a thorough analysis of the results.

**2. Protocol Overview**

**2.1 Control Channel**

The control channel, operating over TCP, is responsible for metadata exchange between the client and server. Let's delve into the key functionalities:

**Metadata Exchange:**

Code Explanation: The client sends the server the file name and size to facilitate synchronization before data transfer.

Code Sample:

# Client-side pseudo code

server\_socket.send(file\_name.encode())

server\_socket.send(str(file\_size).encode())

**2.2 Data Channel**

The data channel utilizes UDP for the actual file transfer. Let's break down its core functionalities:

**Packet Transmission:**

Code Explanation: The server breaks down the file into packets, each containing a sequence number, data payload, and a checksum.

Code Sample:

# Server-side Data Channel Implementation Pseudo code

for packet in data\_packets:

send\_to(data\_socket, packet, 'localhost', 5001)

wait\_for\_acknowledgment(data\_socket)

**Acknowledgment Mechanism:**

Code Explanation: The client acknowledges received packets, ensuring data integrity. Packet retransmission occurs in case of errors or timeouts.

Code Sample:

# Client-side pseudo code

acknowledgment = server\_socket.recv(1024).decode()

received\_sequence, received\_checksum = acknowledgment.split(',')

**3. Out-of-the-Box Thinking**

**3.1 Hybrid Model**

The hybrid model combines the benefits of TCP and UDP. It allows for reliable control channel communication and high-speed data transfer simultaneously. The key implementation is the two-channel architecture, segregating control and data channels:

Code Explanation: Dedicated TCP channel for metadata exchange and UDP for fast data transfer.

Code Sample:

# Two-channel architecture

control\_channel = create\_tcp\_socket()

data\_channel = create\_udp\_socket()

**3.2 Checksum Implementation**

The protocol introduces a checksum mechanism for error detection and correction in the UDP-based data channel:

Code Explanation: Checksums are calculated for each packet before transmission and verified at the receiver's end.

Code Sample:

# Checksum calculation

checksum = calculate\_checksum(data)

packet = f"{sequence\_number:04d}".encode() + data + checksum.encode()

**4. Results and Analysis**

**4.1 Experimental Setup**

The experiments were conducted on a local machine with a multicore processor and 16 GB of RAM. File sizes ranged from 10 KB to 100 MB to comprehensively evaluate performance.

**4.2 Transfer Time and Speed**

Results indicate notable distinctions in performance compared to traditional TCP-based file transfer:

Smaller Files (1 KB): Slight overhead introduced by the control channel impacted transfer time.

Moderate Files (1 MB, 10 MB): The hybrid approach demonstrated reduced transfer time, emphasizing the efficiency of the UDP-based data channel.

Larger Files (100 MB): The hybrid approach showcased a pronounced advantage over TCP, highlighting scalability**.**

**5. Error Handling Scenarios**

The protocol addresses various error scenarios to ensure robust performance:

Packet Loss: The acknowledgment mechanism handles packet loss scenarios, triggering retransmission when necessary.

Checksum Validation: Checksums are used to detect and correct errors during data transfer, ensuring data integrity.

**Timeouts:**

Timeout mechanisms are employed to detect potential packet loss, prompting retransmission, and maintaining protocol stability.

**6. Conclusion**

In conclusion, the implemented hybrid file transfer protocol successfully combines the strengths of TCP and UDP, introducing a novel approach to optimize data transfer mechanisms. The checksum implementation enhances the reliability of the UDP-based data channel. The findings contribute valuable insights into balancing speed and reliability in diverse networking scenarios.

Github Link :