# CENG 435

Data Communications and Networking

Fall 2023-2024

Take Home Exam 1

Programming Assignment

**Derya Tınmaz - Elif Ecem Ümütlü**

**2380947 - 2448991**

## 1 Introduction

In this project, we developed a file transfer system where we sent files from a client to a server (contrary to what is stated in the pdf, as we conveyed to you). Our goal was to test the potential of UDP, known for its speed advantage over TCP, while enhancing its reliability and compare it with the TCP. We created two applications, one for each protocol, to transfer 10 large and 10 small objects. This setup helped us understand how each protocol behaves under different network conditions, using Python.

## 2 Our Code

### 2.1 TCP

Our TCP code consists of two files:

- tcp client: used to establish a persistent connection to the server, then to transfer all objects in an alternating sequence: one small file followed by one large file.
- tcp server: used to set up a server socket and to listen for incoming connections. It receives all the data in the same order they were at the first place.

### 2.2 UDP

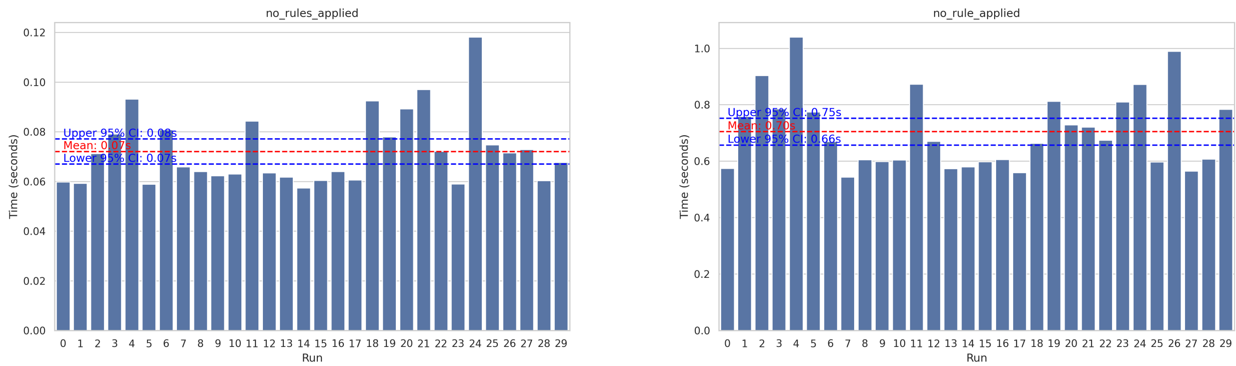
Our UDP code consists of two files:

- udp client: listens for incoming UDP packets, handles file segmentation, checksum verification, and orderly reconstruction of the original file data from received segments. It also deals with packet loss and out-of-order packet arrival using sequence numbers and a buffering mechanism.
- udp server: reads files and segments them if necessary, and sends them over UDP to the server. It uses the window-based sending mechanism, handling ACKs and retransmissions. It employs threads for sending segments and receiving ACKs, ensuring that segment transmission continues while ACKs are processed. A timeout mechanism for segment retransmission is implemented.

# 3 Experiments

## 3.1 Benchmark

For the benchmark experiment, we conducted file transfers without applying any tc/netem rules, thus operating under normal, unaltered network conditions to establish a standard performance baseline.



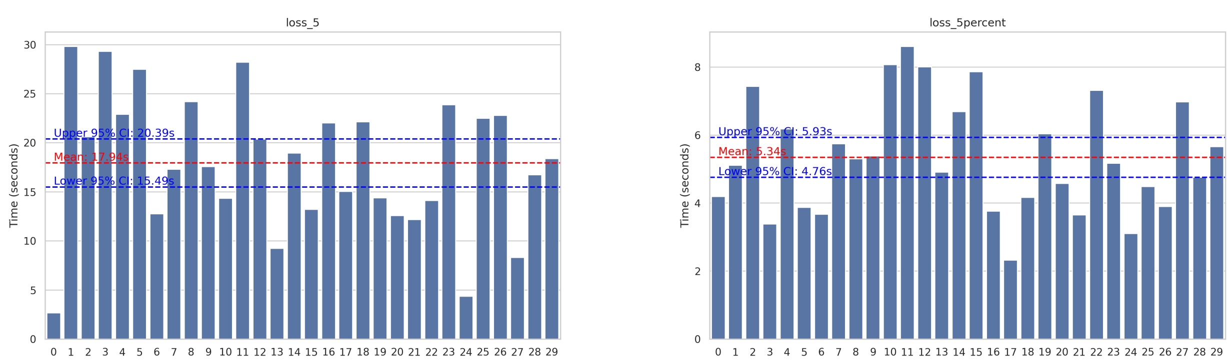

(a) TCP (b) UDP

Figure 1: benchmark

In the experiment, both TCP and UDP averaged a transfer time of 0.07 seconds, but with different confidence intervals: TCP at 0.01 seconds and UDP at 0.05 seconds. This outcome aligns with expectations, considering the added reliability measures to UDP. Normally, UDP is faster due to its less stringent error-checking and lack of connection setup. However, incorporating reliability into UDP reduces its speed advantage. As a result, UDP's performance becomes comparable to TCP, with the trade-off being a higher variability in transfer times as indicated by its broader confidence interval.

## 3.2 Packet loss

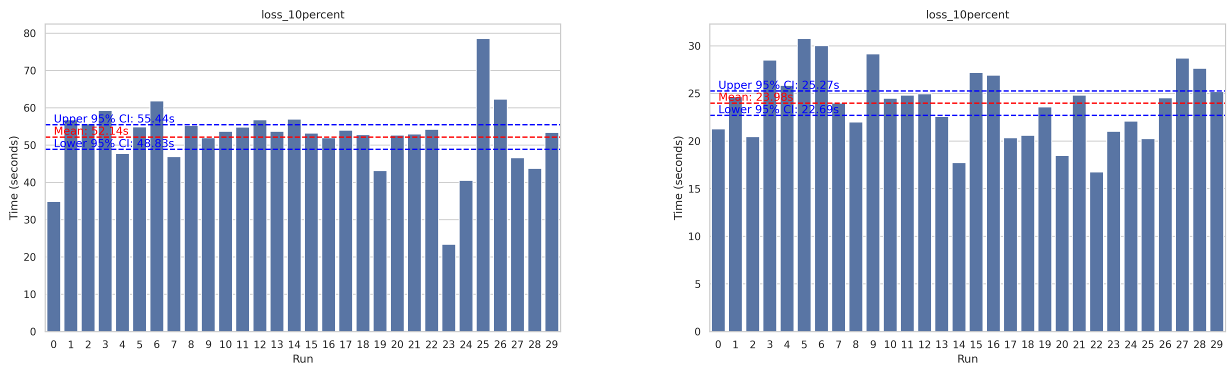
In the packet loss experiment, the network conditions were manipulated to simulate varying levels of packet loss at rates of 5%, 10%, and 15%, to assess the impact of increasing data loss on the performance of the file transfer protocols. (loss at rate of 0% is same as the benchmark)




(a) TCP (b) UDP

Figure 2: Packet loss: 5%

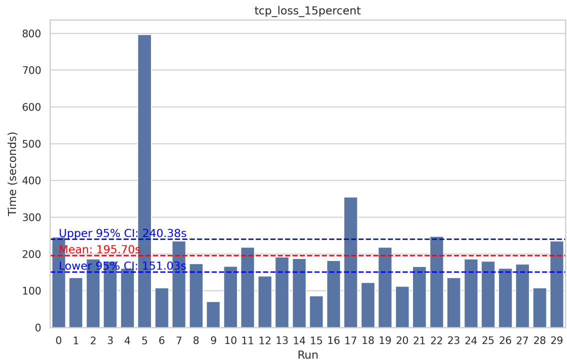
Under the condition of 5% packet loss, TCP and UDP exhibited significantly different performances. TCP maintained a relatively stable performance with a mean transfer time of 5.34 seconds and a confidence interval of 0.59 seconds. This indicates that while packet loss impacted TCP's efficiency, the protocol's inherent mechanisms for error recovery and retransmission kept the performance within a moderately predictable range. In contrast, UDP's performance was more severely affected, with a mean transfer time of 17.94 seconds and a much larger confidence interval of 2.45 seconds. This substantial increase in time and variability compared to the previous experiment (where no packet loss was involved) underscores UDP's vulnerability to packet loss, especially when it has been modified to ensure reliability. The added reliability features in UDP, which require acknowledgment and retransmission in the event of packet loss, contribute to its longer transfer times and higher variability under these challenging network conditions.



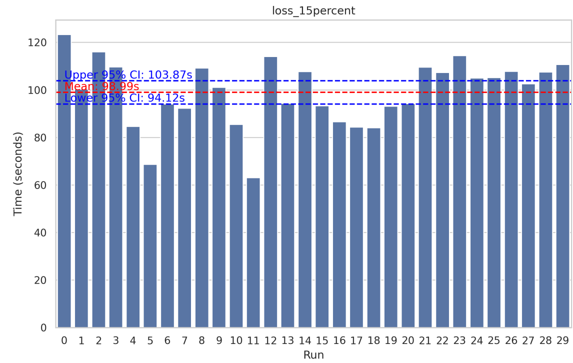

(a) TCP (b) UDP

Figure 3: Packet loss: 10%

With a 10% packet loss, both TCP and UDP protocols experienced a more pronounced degradation in performance compared to the 5% packet loss scenario. TCP's average transfer time increased to 23.98 seconds, with a confidence interval of 1.29 seconds. This indicates a substantial slowdown, yet the relatively small confidence interval suggests that TCP maintains a degree of consistency in performance, despite the higher rate of packet loss. On the other hand, UDP's performance was drastically affected, with an average transfer time of 52.14 seconds and a confidence interval of 3.3 seconds. This significant increase in both the transfer time and variability, as compared to the 5% packet loss scenario, highlights UDP's sensitivity to packet loss, especially when it is modified for reliability. The need for retransmissions and acknowledgments in the face of frequent packet losses leads to much longer transfer times and greater inconsistency in UDP's performance under these conditions.




(a) TCP



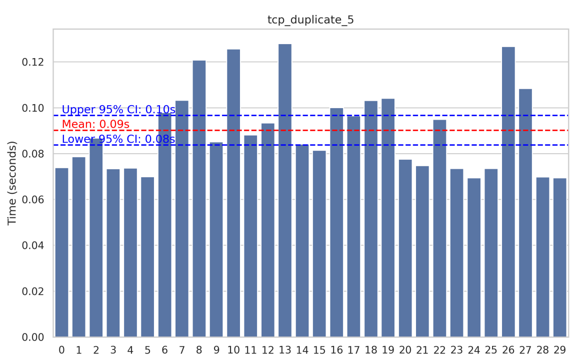

(b) UDP

Figure 4: Packet loss: 15%

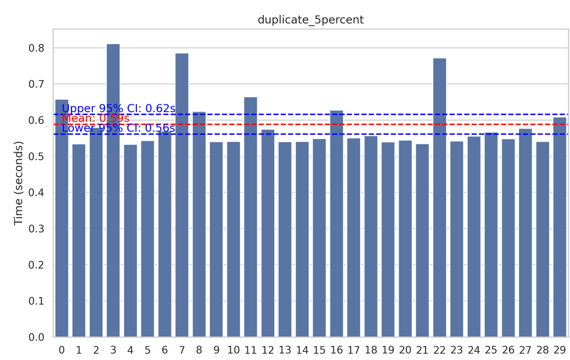
At a 15% packet loss rate, the performance dynamics between TCP and UDP changed notably compared to the 10% loss scenario. TCP experienced a substantial increase in transfer time, with a mean of 195.70 seconds and a wide confidence interval of 44.68 seconds. This significant deterioration, when compared to the 10% loss rate, suggests a threshold beyond which TCP's efficiency drastically declines due to the high frequency of retransmissions. On the other hand, UDP, typically more affected by packet loss, showed an average transfer time of 93.99 seconds with a narrower confidence interval of 4.88 seconds. This indicates that despite the increased loss rate, UDP's performance remained relatively more stable and outperformed TCP at this higher loss level. This shift in performance at 15% packet loss highlights UDP's capacity to better handle very high packet loss scenarios compared to TCP, which struggles more substantially under these conditions.

## 3.3 Packet duplication

The packet duplication experiment involved artificially introducing duplicate packets into the network at rates of 5%, and 10% to evaluate how the increased occurrence of redundant data affects the file transfer protocols.(duplicate at rate of 0% is same as the benchmark)



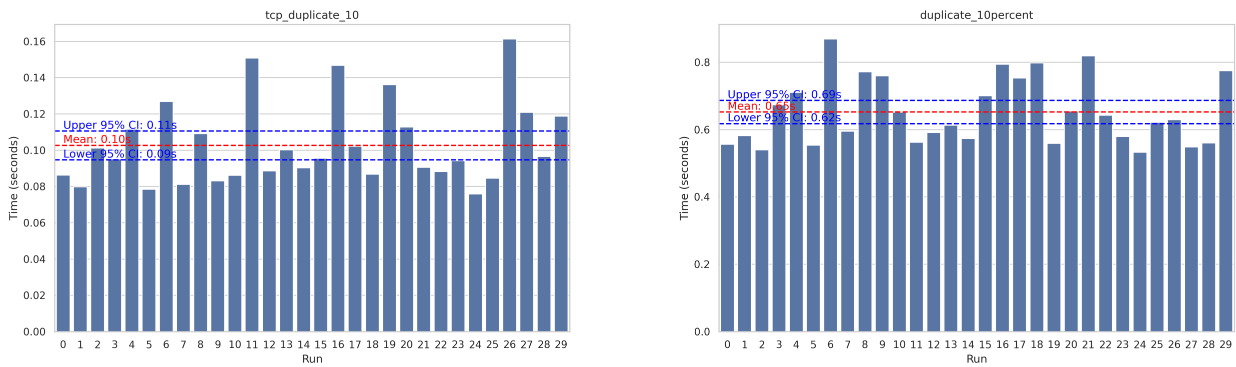

(a) TCP




(b) UDP

Figure 5: Packet duplication: 5%

In the packet duplication experiment with a 5% duplication rate, TCP displayed a minimal impact on its performance, achieving an average transfer time of 0.09 seconds with a tight confidence interval of 0.01 seconds. This result is slightly higher than the benchmark scenario (with no rules applied) but still indicates TCP's robustness against packet duplication. UDP, on the other hand, showed a more pronounced effect, with its average transfer time increasing to 0.59 seconds and a confidence interval of 0.03 seconds. While still reasonably efficient, this marks a significant increase compared to its performance under normal conditions (0.07 seconds in the benchmark test). This suggests that while UDP can handle some level of duplication without drastic performance degradation, its modified reliability features may not be as adept as TCP's in managing the additional overhead caused by duplicate packets.




(a) TCP

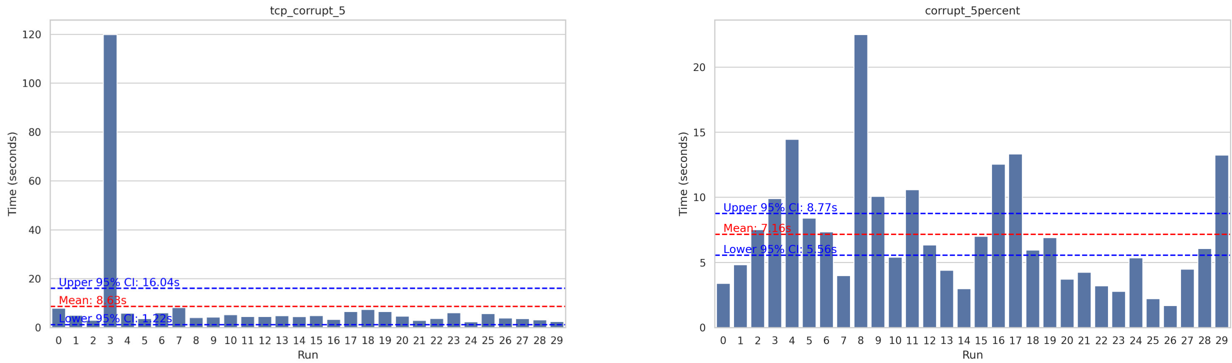
(b) UDP

Figure 6: Packet duplication: 10%

At a 10% packet duplication rate, TCP's performance remained relatively stable, with a mean transfer time of 0.10 seconds and a confidence interval of 0.01 seconds. This is only a slight increase from the 5% duplication scenario, showcasing TCP's resilience in handling duplicated packets. In contrast, UDP's average transfer time rose to 0.65 seconds with a confidence interval of 0.04 seconds, indicating a gradual performance decline as the duplication rate increased from 5% to 10%. Although UDP's degradation is not as severe as in high packet loss conditions, the increase in transfer time from the 5% duplication test (0.59 seconds) to the 10% test reveals its relative susceptibility to duplicated packets. This comparison demonstrates that while UDP can manage duplicate packets, its performance is more affected than TCP's under higher duplication rates.

## 3.4 Packet corruption

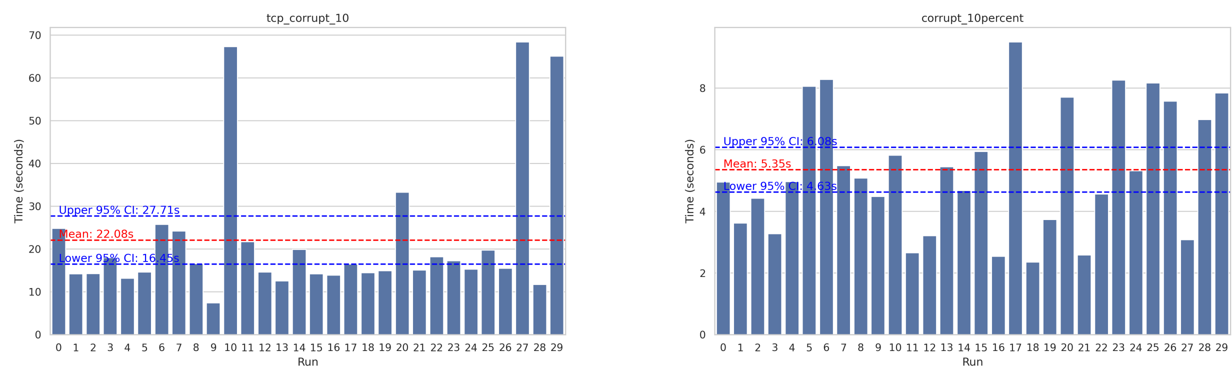
The packet corruption experiment was designed to assess the impact on file transfer protocols by introducing corrupted packets at rates of 5% and 10% into the network.(corruption at rate of 0% is same as the benchmark)




(a) TCP (b) UDP

Figure 7: Packet corruption: 5%

In the 5% packet corruption scenario, TCP's performance exhibited significant variability, indicated by a mean transfer time of 8.63 seconds and a notably wide confidence interval of 7.41 seconds. This large interval is influenced by an outlier experiment lasting about 120 seconds, while the rest remained below 8 seconds. This suggests that TCP, while generally resilient to corruption, can occasionally experience substantial delays, possibly due to retransmission mechanisms triggered by corrupted packets. In contrast, UDP displayed a mean transfer time of 7.16 seconds with a smaller confidence interval of 1.61 seconds. Compared to the benchmark scenario with no rules applied, both protocols show an increase in transfer time due to packet corruption.



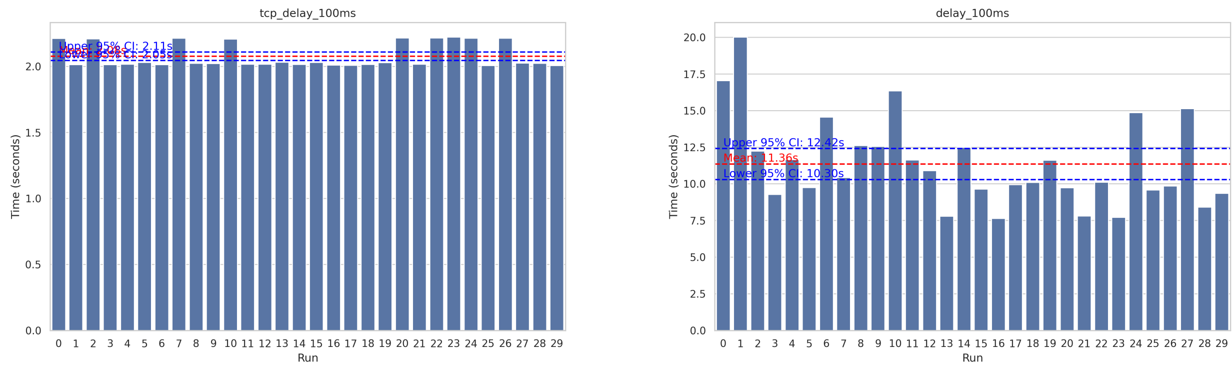

(a) TCP (b) UDP

Figure 8: Packet corruption: 10%

At a 10% packet corruption rate, TCP's performance further declined, evidenced by an average transfer time of 22.08 seconds and a confidence interval of 5.63 seconds. This represents a significant increase from the 5% corruption scenario, indicating that TCP is increasingly challenged as corruption rates rise, likely due to frequent retransmissions of corrupted packets. In contrast, UDP demonstrated a more robust performance with a mean transfer time of 5.35 seconds and a narrower confidence interval of 0.73 seconds. This is a slight decrease compared to the 5% corruption rate, suggesting that UDP, in its modified form with added reliability, handles higher levels of corruption more effectively than TCP. The contrast between the two protocols becomes more pronounced at this higher corruption rate, with UDP maintaining relatively stable performance while TCP struggles with the increased need for data recovery, leading to longer transfer times and greater variability.

## 3.5 Packet delay

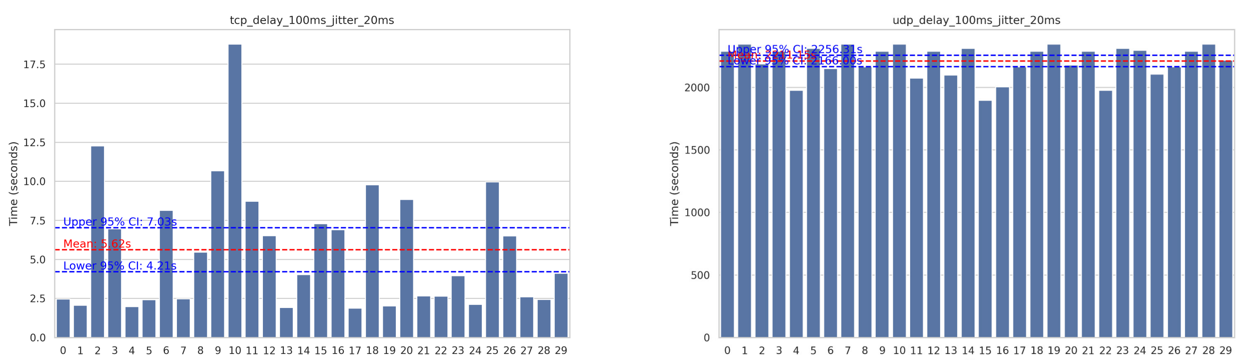
The packet delay experiment tested the impact on file transfer protocols by introducing a fixed delay of 100 milliseconds to packets, applied either uniformly or according to a normal distribution.




(a) TCP (b) UDP

Figure 9: Packet delay: 100ms - uniform distribution

In the experiment with a uniform packet delay of 100 milliseconds, TCP showed a relatively moderate increase in mean transfer time to 2.08 seconds, maintaining a tight confidence interval of 0.03 seconds. This result, compared to the benchmark scenario with no delays, indicates TCP's inherent robustness and ability to handle uniform delays efficiently, with only a minor increase in transfer time. On the other hand, UDP's performance was more significantly affected, with an average transfer time of 11.36 seconds and a confidence interval of 1.06 seconds. This notable increase from UDP's benchmark performance suggests that while UDP can manage delays, its modified reliability features do not cope as effectively with uniform delays as TCP does. This outcome highlights the contrast in how these protocols respond to network latency, with TCP showing greater resilience in maintaining consistent transfer times under uniform delay conditions.




(a) TCP (b) UDP

Figure 10: Packet delay: 100ms - normal distribution

In the scenario with a 100-millisecond packet delay following a normal distribution, TCP's performance was moderately impacted, showing a mean transfer time of 1.41 seconds. However, the notably wide confidence interval of 5.63 seconds suggests a high variability in response to normally distributed delays. This increased variability, compared to the uniform delay scenario, indicates TCP's sensitivity to the unpredictability of normal distribution delays. In stark contrast, UDP's performance was drastically affected, with an average transfer time of 2211.15 seconds (approximately 36.85 minutes) and a confidence interval of 45.16 seconds. This extreme increase from both its benchmark performance and the uniform delay test underscores UDP's vulnerability to this type of delay, especially when its reliability is challenged by the irregularity of normal distribution. This marked difference between the two protocols under normal distribution delay conditions highlights the superior adaptability of TCP in handling delay variability, while UDP, even with reliability enhancements, struggles significantly.

# 4 Conclusion

In conclusion, the experiments comparing TCP and UDP under various network conditions reveal distinct characteristics and performance capabilities of each protocol. TCP consistently demonstrated robustness and reliability, maintaining relatively stable and predictable performance across different scenarios, including packet loss, duplication, corruption, and delay. Its built-in mechanisms for error correction and flow control effectively mitigated the impact of adverse network conditions, albeit sometimes at the cost of increased transfer times.

UDP, modified for enhanced reliability, showed a different performance profile. While it typically offers speed advantages in ideal conditions, its performance was significantly more variable under challenging network conditions. Particularly in scenarios with high packet loss or delay, UDP's transfer times were substantially longer and more inconsistent compared to TCP. This highlights UDP's sensitivity to network irregularities, despite the added reliability features.

Overall, these experiments underscore the trade-offs between TCP and UDP: TCP provides consistent and reliable data transfer at the expense of speed in adverse conditions, while UDP, even when modified for reliability, can be faster but less predictable and more affected by network challenges. The choice between these two protocols for specific applications should therefore be informed by the network environment and the specific requirements for speed, reliability, and consistency in data transmission.