Homework 1 - OLARU BIANCA, MISS 1 2

This document discusses the implementation of a client-server program for handling data transfers using TCP and UDP protocols and 2 transmission mechanisms, streaming and stop-and-wait. For increased security, the system also uses SSL/TLS encryption for TCP communication, as it is not secure on its own. To account for some challenges and fallacies that distributed computing presents, the system was designed with fault tolerance and acknowledgement mechanisms.

To ensure reliability, the stop-and-wait mechanism checks that each message is acknowledged before sending the next one, while timeouts and retries are implemented to account for any network delays. For a flexible program, the client can handle reconnections and adapt to changes in the network, and users can specify custom server addresses and ports. To prevent unauthorized access, SSL encryption is added to TCP connections through Python’s ssl module.

Testing was performed for all combinations of protocols and mechanisms,

and the main issue appears to be during TCP streaming connections: the server only receives around 80 messages out of the 1024 sent, while the number of received bytes is equal to the ones sent. It can be assumed that multiple messages are being combined into one before arriving at the server because of the way TCP deals with message boundaries. Time-wise, both protocols performed well with the streaming mechanism with a time of 0.003s. For stop-and-wait, UDP was the faster one with a time of 0.035s, while TCP completed the task in 0.041s.

TCP streaming - 0.003s, 80/1024

UDP streaming - 0.003s, 1024/1024

TCP stop and wait - 0.041s, 1024/1024

UDP stop-and-wait - 0.035s, 1024/1024

Regarding future improvements, the TCP message loss can be addressed by using message length encoding. Sending the message length along with the message would ensure the server would better recognize message boundaries. Another issue is the UDP server not closing automatically, for which a termination condition should be implemented.