PORTFOLIO 2

DATA2410 – OSLO metropolitan university

FAHMI HAYBE MOHAMMED / s362106, BEHDAD NIKKHAH / s362085, TERESA PHAM / s345368, JAN NICOLE RUBIC YAO / s362049

2023

TABLE OF CONTENTS

[1. Introduction 2](#_Toc133855322)

[2. Background 2](#_Toc133855323)

[3. Implementation 2](#_Toc133855324)

[4. Discussion 2](#_Toc133855325)

[5. Conclusions 2](#_Toc133855326)

[6. References (if any: mention sources) 2](#_Toc133855327)

# 1. Introduction

An introduction should tell the reader why this work is interesting.  
It should describe:

This report contains an implementation of a (DRTP)- data reliable transport protocol. This program is built in python and is using socket module with UDP to transfer data. Since UDP is an unreliable protocol, DRTP would ensure a reliable connection. The network tool “Mininet” is also an important tool to build a virtualized network regarding the simple topology. This will be useful to evaluate the virtualized network resources and performance when using DRTP. (Include the bonus as an implemented work)

The test would be done by the custom written code to transfer a file, and to measure the throughput using different RTTs (round-trip-time). It would be performed different various of function to trigger retransmission of packets either lost or skipped. The performed tests should reveal if the code is handling potential losses, reordering and duplicates.

The approach to solve the problem is to implement a (three-way handshake) between the client and server. Each reliable functions have its own client and server methods.

Include limitations here!!!

This document is structured in five segments, including the list of reference at the end of the document. The first chapter would start by presenting the background, containing appropriate theoretical instructions. The next chapter is the implementation of the custom/customized written code, and what sort of functions used. The discussion of the different test performed would be discussed in the next following chapter, also following the result of the tests. The last’s chapters would subdue a conclusion and list of sources.

# 2. Background

Your background section should include the appropriate theoretical background that your reader should know before delving into the details. For example, what is stop and wait? How does it work?

# 3. Implementation

# 4. Discussion

1. Run the file transfer application with stop\_and\_wait reliable protocol, GBN with window sizes 5, 10, 15, and GBN-SR with window sizes 5, 10, 15 using RTTs 25, 50 and 100ms. Calculate

throughput values for all these cases and explain your results.

|  |  |  |
| --- | --- | --- |
| Stop\_and\_wait | RTT | Output |
| Test 1 | 25ms | Successful received the file (saved in test-1-saw) |
| Test 2 | 50ms | Successful received the file (saved in test-2-saw) |
| Test 3 | 100ms | Successful received the file (saved in test-3-saw) |

1. Write a test case to skip an ack - this will trigger retransmission. Test with all three reliable functions.
2. Write a test case to skip a sequence number to show the out-of-order delivery effect. This will also trigger retransmission. Test with GBN, and SR. Report your results.
3. Use your artificial testcases to show the efficacy of your solution. Your solution should be able to handle losses, reordering and duplicates.

Test Case Name: Test RECV\_STOP with skip\_ack=True

Test Case Description: Test the RECV\_STOP function for skipping the first acknowledgment message, which should trigger retransmission at the sender side.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST ID | Scenario | Test Step | Expected Result | Actual Outcome |
| SKIP\_ACK\_SAW | Verify that retransmission occurs when first ACK message is skipped | 1. Run the application.py program in server/receiver mode with the optional flag -t skip\_ack 2. Run the application.py in client/sender mode with the same reliable method as the server/receiver. 3. Verify that the receiver skips sending the first ACK message. 4. Verify that the sender retransmits the packet | * The server/receiver should skip sending the first ACK message. * The client/sender will retransmit the previously sent packet because of timeout | * The sender retrasmitted the packet as expected after timeour |
|  |  |  |  |  |

# 5. Conclusions

# 6. References (if any: mention sources)