RELIABLE TRANSFER PROTOCOL DESIGN DOCUMENT

Group 1
Waquar Wasif: 2018A7PS0254H
Tulaib Ahmed Abdullah: 2018A7PS0272H
Ritika Reddy: 2018A7PS1224H
Anvitha Nallan: 2018A7PS1214H
Abhishek Mishra: 2018A7PS0019H
Navdeep Singh : 2017B5A71675H

Group 2
Anurag Aribandi: 2018A7PS1218H
Varun Narayanan: 2018A7PS1226H
Himnish Kapoor: 2018A7PS1215H
Anish Mitta: 2018A7PS1221H

Abstract

We are implementing a form of file transferring protocol which allows a host to **send and receive** data over a server with another host. This protocol will use UDP sockets to communicate. This document describes the protocols, structures and commands required for data transfer.

This protocol uses block numbers to identify packets and opcodes to identify the command associated with it. We use acknowledgements to determine whether the packet has been delivered successfully to the clients buffer and negative acknowledgements otherwise. Packets are retransmitted until an acknowledgement is received or until a predefined timeout duration.

Methods and Commands

Below we have defined the methods and commands to be used in tandem with the transport layer and application. This is a tentative definition of what they do, their arguments and what they return.

//We faced some difficulties implementing delete and store commands and hence they have been removed

a) **Initialize (RINIT) : //similar to configuration**
   Server address and port details are bundled and saved for initialization of the connection.

b) **Send (RSEN) :**
   (in case of server:server receives send request and then sends packets until eof)
   (in case of client: when it wants server to send file it sends a send request, ack is sent after each data packet is received from the server)

Send data to the server in frames in a defined structure and wait for acknowledgement from the server.

c) **Acknowledge (ACK) :**
Sends an acknowledgement packet to the specified address and port. Will contain the block number of the packet being acknowledged.

d) **Receive (RREC) :**
(in case of server: server gets a receive request and sends ack and then starts receiving data and writing it a new file)
(in case of client: when client wants server to receive a file it sends receive request and then after receiving an ack it starts sending data to server.)
Uses UDP socket to receive data from the server in a receive buffer from specified address and port.

Packet Structures

Below we have defined the packet structure for the commands which require it as well as an error packet. Operation codes are assigned to each command for identification purposes.

NACK/ACK

String	Int	Int
Opcode	Block #	Ack or Nack

RDEF

String	string
Opcode	Pathname

RSTF

String	string	Object
Opcode	Pathname	Data

RSEN

String	Int	Object	String
Opcode	Block #	Data	Pathname

RREC

String	2 bytes	String
Opcode	Block #	Pathname

ERROR

Int	String
Err Code	Error Message

Reliability and Network Conditions

The protocol will implement a Selective Repeat Paradigm for data transmission and for handling packet loss. For each packet, the server will send an ACK packet back, if the client receives a NACK packet back or does not receive an ACK packet that means that there is a block number missing in the data packets constituting the total data and it is considered to be lost. To handle this, if a NACK packet is received by the client, it will retransmit the block number specified in the NACK packet, ensuring no packet loss overall.

In the case of poor or slow network connections, if an ACK packet is not sent before a defined time period expires, then the packet will be retransmitted again regardless. These will be implemented to ensure reliable data transmission as UDP does not guarantee that.

Errors

If at any point there is an error, there will be an error displayed on the client side or packet transmitted to the server side if required. The errors handled by our protocol will be as follows :

Illegal or Invalid Opcode Error : If the opcode in a packet does not exist or is invalid

Out of Bounds Block Number Error : If the block number for a data packet is not valid or out of bounds.

File or Data Not Found Error : Primarily for the send and delete commands, if there is no data/file to send/delete then this error will be triggered.

Initialization Error : If an error occurs during the initialization of the sockets or with the server address and port number, this error will be triggered.