Project 2

Simple Window-based Reliable Data Transfer

CS 118 Computer Network Fundamentals
Fall 2015

Alfred Lucero

ID: 604251044, SEASnet Login: alfred

Khoa Nguyen

ID: 204329993, SEASnet Login: khoanv

Overview

First, we implemented basic UDP file transfer and tested sending image files, assuming there was no chance of loss. Then, we added probability inputs and began to use duplicate ACKs and timeouts to simulate packet loss and corruption with certain chances and retransmission to ensure reliable data transfer.

At first, we experienced difficulties with implementing the probabilities of loss and corruption, but we ended up using the aid of rand() function based on inputs to determine whether or not the sender and receiver would receive an ACK or data file properly. The select() function aided with timeout retransmission, and we incrementally developed the rdt over UDP to model TCP's implementation with socket programming principles.

Basic UDP Implementation

Packet Struct/Header: holds flags (ACK, REQ, DAT, FIN), sequence number, data size, and data array (1024 bytes)

Socket Functions: used protocol methods such as socket, sendto, recvfrom, and close with parameters such as SOCK_DGRAM

Go-Back-N implementation

Packet Loss: used select() function and timeout value of 1s to handle timeout and retransmission

Packet Corruption: use duplicate ACK algorithm to handle corruption for retransmission

Test Cases:

Set-up: make (use this in directory of the files of UDP rdt project)

Sender: shell > sender < portnumber > CWND PL PC

Receiver: shell > receiver < sender_hostname > < sender_portnumber > < filename> PL PC

Sender side: ./server 4444 4 0.0 0.0

Receiver side: ./client 127.0.0.1 4444 test.jpg 0.0 0.0

With packet lost and corruption accounted for:

Sender side: ./server 4444 4 0.1 0.5

Receiver side: ./client 127.0.0.1 4444 test.jpg 0.1 0.1

EXAMPLE 1: Basic File Transfer over UDP

./server 4444 4 0.0 0.0 -> ./client 127.0.0.1 4444 test.jpg 0.0 0.0

EXAMPLE 2: 100% Packet Loss at Receiver

./server 4444 4 0.0 0.0 -> ./client 127.0.0.1 4444 test.jpg 1.0 0.0

EXAMPLE 3: Large File with Some Loss and Corruption Probability

./server 4444 4 0.1 0.5 -> ./client 127.0.0.1 4444 large.zip 0.1 0.1

EXAMPLE 4: RDT Procedure Analysis over Small File Transfer

./server 4444 4 0.25 0.5 -> ./client 127.0.0.1 4444 test.jpg 0.25 0.5

// Go over Go-Back-N retransmissions due to timeouts and duplicate ACKs in this case with

// probability of packet loss and corruption at both sender and receiver

File Integrity:

diff receivedFile test.jpg // outputs nothing if no differences between the file sent/received