FCN Project 2

Anil Poriya

arp6763

Introduction

The application allows the user to do a reliable data transfer (Image or Sensor data) over UDP by mimicking the working of TCP. Since congestion control will not be an issue in this case (to be tested against one sender and receiver), the application does not provide the congestion control mechanism. The application uses fast retransmit algorithm. The project contains the following java files

1. Basestation
2. Buoy
3. Controller
4. Image_Message

Maximum Segment Size taken → 1019 bytes.

Message Format

Sequence Number
ACK Number
Control Field
Checksum
EOF
Data

Image_Message Objects

The message objects will contain the following fields:

1. Data

This is basically the byte array which will store the some chunk of original data. Each byte array of data taken is 1019 bytes long. We take this class to basically form the initial chunks and store them in an arraylist so that they are easily resent whenever required.

Basic Flow of the Application

The Basestation will request either Image or Sensor data from the distant buoy. The first packet from base station will basically contain a single byte indicating the request(whether need sensor data or image or close this communication). The buoy on receiving the request from base station will then send sensor data or image back to the base station. Once everything has been done, base station will then simply tell the buoy that it has got everything it wanted and would now like to close this communication.

Image Transfer:

The buoy will first transmit a packet stating the size of the image byte array to the base station. Thus the first packet will contain a 4-byte array stating how much long the image byte buffer should be at the base station to receive all the packets.

On receiving this packet, base station will create a buffer equal to the size specified.

The buoy transmits the packets according to the window size. If the packet or data to be sent is larger than the maximum segment size, then the data is split into different chunks and stored with appropriate sequence numbers. The sequence numbers are stored in the first 4 bytes of the array. This is followed by one optional byte stating whether the current packet is the last packet. If it is the last packet, then we send a 1 as that byte to indicate that the last of packet has been received.

Additionally, since checksum happens in UDP under the hood, we do not include the concept of checksum in our protocol because since it rarely happens.

If the packet to be sent is the last packet, the client will mark this packet as the last packet by setting its 5th bit value as 1. These packets which are stored as "messages" are sent by buoy one by one.

First 4 Bytes(Sequence Number)	5th Byte stating whether it is last byte.	1019 Bytes of actual Image data.

1024 Byte array.

The base station will receive this packets one by one. Since packets rarely drop and checksum will only fail if packets are corrupt(again rare scenario), we simply send an acknowledgment back to the buoy once the packet is received. This acknowledgment packet is 4-byte long.

If the buoy does not receive an acknowledgment when it sends a packet within 1000 milliseconds (Timeout Period), it will resend the packet.

Note: At the buoy side, we have maintained a boolean array of size -> number of packets. This array indicates whether a particular packet has been successfully received by the sender(through acknowledgment). At every stage we check this boolean array. If any of the value in boolean array is false, we send packet with that sequence number again to the receiver. This helps us in achieving **SELECTIVE REPEAT** of packets rather than sending all packets back to the receiver.

The protocol allows us to send both Image files and Sensor data based on what Basestation requests. For Images, the user will enter the location of the image. The image will then be transferred to the receiver end where upon reordering the packets, an image file will be generated at the receiver end. The details for running this application has been given in **Run_Notes.txt** file.

Case 1. Delay

If the delay is more and it takes more than 1000 seconds, the sender would retransmit. At the client side the duplicate packet is ignored.

Case 2. Packet

If there is a timeout, the client would retransmit the packet.

Case 3. Reordering

At the receiver side, it checks whether the packet received is valid. The packets may arrive in any order. If the checksum matches, the receiver will just accept the packet and store it an appropriate location of sequence number. This helps us in reordering of packets easily as we just have to iterate the ArrayList.