Yuanqi Li, 504759910, lyuanqi@cs.ucla.edu
Zhengkai Zhang, 604582162, kirkzhang49@gmail.com

Project 2: Implementing Window-based Reliable Data Transfer Protocol

Abstract

In this project, our goal is to implement a window-based, reliable data transfer protocol with selective and repeat protocol. Our project is written in C++ using User Datagram Protocol socket which is not a reliable data delivery protocol. There will be a client program ask a file from server with reliable delivery in our project.

Selective and Repeat protocol

Both sender and receiver side of selective repeat protocol has a window of sequence numbers. It is very similar to Go-Back-N protocol while it only retransmits the frame which is damaged or lost. The selective repeat protocol, a packet can be received out of order and they are sorted to maintain a proper sequence of frames. If packet is skipped, the receiver part will recognize that packet is lost and send ACK to sender. Sender will search the packet in its window and retransmits that frame. Receiver also will sorting the frame in proper sequence as it receives the retransmitted frame whose sequence is out of order of receiving packet. Overall, the selective repeat protocol won't wast bandwidth for packet that are received in correct order. It is more efficient than Go-Back-N protocol, but it is more expensive and complex.

Project Implementation

Packet

We defined the package information and function with in packet.h file and packet.coo file. We defined eight type of packet type which are DATA, ACK, SYN, SYN_ACK, FIN, FIN_ACK, MSG, and REQ within an enum function. We have packet type, sequence number, filesize, wrap_count flag, buffer size, and a buffer to store our packet data. All the packet type will be apply to our implementation, and we defined proper functions to handle these data in packet class. Our initial packet size is 1024, and our data size is equal to (PACKET_SIZE - sizeof(PacketType) - sizeof(unsigned long) - 3*sizeof(unsigned int)).

Window and Buffer

We defined a class of WindowElement, and a class window using vector to keep these elements as our window based model. These classes are defined in window.h and window.cpp file. The WindowElement include a boolean value for check start, timestamp, status of sent, not sent, resent, and acked, and our packet class as the data contain in the WindowElement which defined above. WindowElement class have proper functions to get or set the time, packet or status of WindowElement. The window defined using C++ data structure to store elements in side the window and pending_packet. Inside of window class, we defined our functions to handle time out, get next window element to sent, remove acked window element, fill window and check each status of our current window.

Utils.h and utils.cpp files are the files we defined to handle the file and buffer status check, read and write from buffer to file, reset and get string from file, and exist with proper error print. These files are defined here to reduce the file and buffer manipulating code in our server and client file. MAX_SEQNUM, WINDOW_SIZE_BYTES, and TIMEOUT_MS are default value given by instructor which defined in our setting.h file.

Some Functions Definition

WindowElement::setStartTimer()
WindowElement::getTimeElapsedMs()
Window::getTimedOutWindowElement()
These functions are designed to set the timer to start and getTimer elapse of our window elements which help our project to handle the time out part.The get timed out window element function will return the element which are time out for resent.

WindowElement::getPakcet()
WindowElement::getStatus()
WindowElement::setStatus
WindowElement::WindowElement
These functions are the part to set up our WindowElement, get the packet, status, and set the

status. WindowElement function set the default value for our window element type as Not_sent, and HasStart value false. Assign the packet to our WindowElement.

Window::fillWindow()
FillWindow function will put all elements inside of our pending_packets to the elements vector defined in our window class. It will pop_back the packet inside of pending_packets and push_back into our elements vector until our pending_packets become empty or the windows get to it's packet_capacity.

Window::hasUnsentPacket()
Window::getNextWindowElementToSend()
Window::removeAckedWindowElementBySeqnumAndWrapCount()
These two functions are defined here to check if there are an Unsent packet which check all element in our elements vector return the packet which status is UN_SENT. The hasUnsentPacket function will return it as a boolean value while getNextWindowElementTo send will return the window element which is un_sent type.
The remove Acked Window Element will help the server part to remove acked window element inside of our current window element vector by check the sequence number and the wrap_count flag.

Client and Server

Server
The server part of program is defined with server.cpp file. It start with ./server with a port number. It will setup a UDP socket and bind the port number to our UDP socket and start to waiting for the request from client part.
There are three functions defined here to help server divide packet from file and sent the packet to client.
getNextSeqnum
splitFileToPackets
sendPacket
The getNextSeqnum will help to get the next sequence number for our packet sending. The splitFileToPackets will receive a filename and read all content to a buffer divide the buffer account to packet default size and push_back them to an all_vector packet, and return this vector. The sendPacket function will check the type of packets and print the proper message and send the packet to client using sendto function.

In the Main function, server program will looping and receive packets from client program. It will check each packet type and go to a switch case to handle all different type of packet sent from client program. For the REQ type, it will open the file and split the file content to packets. We defined SYN and FIN type to handle starting and ending of the data transfer. The server program will output correct message and send back the ack to client program, it will handle all the retransmission case and time out case. The server program will not stop when a data is safely translated, it will wait the next client request another file or exist by user manually.

Client
Our client program define in client.cpp file which will request a file from server program. The client will take three arguments from user which is hostname port number and request file name. We defined a init function to start the program to set connection with server program. We defined some functions to help our client program.

combinePacketsToFile
sendPacket
isDuplicatePacket

The combinePacketsToFile function will sort all packets by wrap_count and sequence number.Then the function will memory copy all the packet to a buffer, and it write the buffer using writeCharArrayToFile to write our data to the output file. SendPacket function is almost the same as send packet function in our server program. It will using a switch case to output the correct message for each data type case and send it to Server program. The is duplicate packet function will check the all packet to see if there is a duplicate packet has the same sequence number and the same wrap_count and return a boolean value.

In the main function, client program will prepare a SYN packet to ask server program for a request file. It will keep looping receive the packet from server and handle all packet type with the switch case, and it will send the correct ack packet as a signal to server program tell it client have got the correct packet in order or not. It will packet all window element into a buffer in correct order and write them to an output file. At the end of transform data process, it will receive a FIN from server and return a FIN-ACK to server. Until the process done with three way handshake the client program will exist.

Difficulties Faced

Window Class Design

It takes a mount of time for use to design the Window to handle the selective and repeat protocol algorithm. At first, we use C as our language and find it is a little harder the come out with a Window implementation in C, so we switch the program language to C++.

The C++ program has the nice vector data structure which is a prefect fit to our Window element base

packet with selective and repeat algorithm. The sorting and unlimited size feature is good fit for our window class design. The C++ vector store all our pending packet and current window packet in two different vector in our window class, so we can properly handle all the requirement for lose packet and time out of selective and repeat protocol algorithm.

Output File

There is a small bug when we try to output out data to the output file. We found the binary file we translate from server to client can not be properly write. We search online for the reason, and we eventually found the reason is we use c++ string copy api to make the buffer for our output file. After read the resource from online, we decide to use memcpy() function to handle our output file. After all, our program handle all type of files include image, pdf, and etc.

Large file handle (large than 30KB)

This was the most difficult part we found when we were debugging our project. In default, our MAX_SEQNUM set to 30720 which can only handle up to 30 KB for our file size. When we test large file which large than 30KB, our program fail to handle it.For this case, we use a wrap_count flag to handle the large file case. At the time, our sequence number large than our default MAX_SEQNUM, our sequence number will be set to zero and wrap_count will increase by one. We insert this flag all over our project and eventually fix the bug of translate the large file.

Timeout Implementation

The time out feature is quit hard, we need to found the proper API to handle the time out feature. We using the clock_gettime API to help us to check for the time out cases. In the WindowElement::getTimeElapsedMs() function, we have to calculator the tv_sec and tv_nsec to milliseconds and translate it to a int format. At the end, we were able to handle the time out feature of the selective and repeat protocol algorithm.

TEST CASE

For the test case, we request 6 different file with 2 main different type(binary file and txt file). Each type has three level of file size:
small(5bytes) ,middle(5bytes - 30kb) ,and large(above 30kb). We were using a client program in another folder to request the file from our server program folder. For result, we check files using diff (- - text option for binary file) to test our request output file is identical to the original file.

For packet lost, we use tc command change the packet lose rate to 10%,30%, and 50%. We check the output of our program to check it fit to the selective and repeat protocol algorithm pattern.

Summary

Overall, the project is very helpful for use to understand the every detail of selective and repeat protocol algorithm. We designed the structure of our window based reliable data transfer using UPD socket from empty. Our implementation divide the project to separate modulation which are easy to understand, debug, and modify the feature in future. There are number of helpful APIs applied to our program which we have learned and applied to this project.