CS 118 Computer Network Fundamentals
Project 2: Simple Window-based Reliable Data Transfer in C/C++

Team Member 1

Name: Darin Minamoto

Student ID: 704140102

Team Member 2

Name: Lingyu Zhang

Student ID: 404205755

1. Design Description

1) Header Format

```
int type;              // Message type: REQUEST/DATA/ACK
int seq_no;            // Sequence number
int contentLength;     // Size of data content in bytes
int ack;               // ACK number
int fin;               // FIN flag
char data[DATA_SIZE];  // Actual data content: 1KB maximum
```

We choose not to implement checksum and use a probability parameter to simulate packet corruption as suggested in the spec. Therefore, the header is 20 bytes in total.

2) Messages

We define 3 types of messages (`REQUEST/DATA/ACK`) indicated by the `type` field in the header.

i) `REQUEST`

Used by the client to request files from the server. The data content of a `REQUEST` message should be the requested filename.

ii) `DATA`

Used by the server to send the requested file to the client. The server breaks up the file into 1 KB packets and sends the packets in order indicated by the sequence number field in the header. It also uses the ACK number acknowledge the ACK packets sent by the client.

iii) `ACK`

Used by both the server and the client to acknowledge packets with the proper ACK number. The data content of an `ACK` message is empty.

3) Timeouts

We choose to implement the Go-Back-N protocol to deal with retransmission due to packet loss and corruption.

For the server, there are 2 types of timeouts and we use a state variable to differentiate them.

i) Timer for the sliding window

To implement the Go-Back-N protocol, we set a timer for the first packet in the window. The timer starts when the first packet is sent and times out if no ACK is received for the packet or the received ACKs are corrupted. If timeout occurs, the server retransmits all packets in the window.

ii) Timer for [FIN, ACK]

After the server sends a [FIN, ACK] packet to acknowledge the [FIN, ACK] sent by the client, it should wait for possible resends of the [FIN, ACK] by the client to ensure the client receives it. If the timer times out without receiving duplicate [FIN, ACK]s, the server can safely terminate the connection.

For the client, we set a timer for the [FIN, ACK]. A timeout indicates that the [FIN, ACK] packet from the server is lost or corrupted and the client needs to resend the [FIN, ACK]. We choose not to set a timer for the initial request message because it is not important for this project according to the TA.

4) Window-based Protocol

The `rdt_window` class implements the Go-Back-N sliding window. It keeps a list of pointers to the packets in the window and the current sequence number of the file. We explain some of the key functions below:

`void rdt_window::slide_window()`

Slides the window past the packet acknowledged by the last received ACK with the highest ACK number. Note that the server can safely slide the window past all the previous packets even if the ACKs acknowledging previous packets are lost because if the server receives an ACK with a higher ACK number, the client has already received all the data up to that higher ACK number.

`vector<char*> rdt_window::fillWindow()`

For all the empty slots in the window, create new packets from the file and send them.

For the client, it keeps the last in order packet `rdt_packet last_seqpacket` to check whether the next received packet is in order based on the sequence number. If it receives a packet that is corrupt or out of order, it resends the ACK packet for the last in order packet it received and drops the out of order packet.

We make a special case if the very first data packet is lost. The client does not send ACK packets with ACK# = 0 and Seq# = 0 for the subsequent out of order packets. Instead, it waits the server's window timer times out so that the server resends all the packets in the current window.

5) Connection Termination

We decide to let the server send the initial FIN packet to close the connection after it finishes sending the requested file.

2. Problems and Solutions

1) ACK# and Seq#

At first, we had difficulties figuring out how the ACK number and sequence number work and later solved it by carefully reading the book.

2) Deleting Packets

In our implementation of the sliding window, we use pointers to keep track of packets. But when we first attempted to delete packets that are slid past by the window, we had memory management problems because other classes might still be using the packet. We solved it by keeping a dependency list of all the packets and delete only those that no one has reference to.