RELIABLE USER DATAGRAM PROTOCOL

SR. NO.	GROUP NO.	NAME	ID
1	1	MANAN PATEL	2018A7PS0194H
2		THAKKAR PREET	2018A7PS0313H
3		KEVIN SHAH	2018A7PS0375H
4		ADIT GANDHI	2018A7PS0575H
5	2	ACHYUTA KRISHNA V	2018A7PS0165H
6		ANIRUDH G	2018A7PS0217H
7		AJITH P J	2018A7PS0040H
8		J ALVIN RONNIE	2018A7PS0029H
9		MANEESH REDDY	2018A7PS0462H

Introduction

This Reliable User Datagram Protocol is designed to provide a data transport service for file transfer ensuring reliability in environments with data loss, transmission delay and out-of-order delivery of data.

Background

The RUDP defined here caters to applications which cannot handle change in TCP congestion window but requires reliable delivery. The protocol uses UDP as the transport protocol and builds reliability in the application layer.
The protocol should meet the following criteria:

1. It should provide in-order delivery.
2. It should be a sequenced packet based transport protocol.
3. It should provide persistent connection.
4. It should provide error detection.
5. It should handle data loss during transmission with retransmissions.

Data structure format

UDP Header:

< 16-bits >< 16-bits >

SOURCE PORT ADDRESS	DESTINATION PORT ADDRESS
LENGTH	CHECKSUM

RUDP Header:

Mandatory fields of the header:

1. Flags :
    a. ACK is set if it is a ACK segment
    b. FIN is set if it is a FIN segment
    c. SYN is set if the packet contains SYN segment
    d. DATA is set if the packet is a DATA segment
    e. NUL is set if the packet is empty
2. SEQUENCE NUMBER: A number specific to a packet
3. ACKNOWLEDGEMENT NUMBER: The sequence number of the last received segment.
4. HEADER LENGTH: Size of the header in the segment.
    - Segment length - Header length = Size of the data transmitted.

Different Segments

1. Data Segment: The data segment of the UDP packet contains RUDP header and its data in the following format.

<8-bits><8-bits>

SYN	ACK	FIN	DATA	NUL	0	0	0	HEADER LENGTH
SEQUENCE NUMBER								ACKNOWLEDGEMENT NUMBER
DATA								

2. Acknowledgement (ACK) segment : Used to acknowledge a data segment.

<8-bits><8-bits>

0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SEQUENCE NUMBER								ACKNOWLEDGEMENT NUMBER							

3. Close (FIN) segment : Segment to be used for closing the connection

<8-bits><8-bits>

0	ACK	1	0	0	0	0	0	0	0	0	0	0	1	0	0
SEQUENCE NUMBER								ACKNOWLEDGEMENT NUMBER							

4. Null (NUL) segment : Segment used to keep alive connection

<8-bits><8-bits>

0	ACK	1	0	0	0	0	0	0	0	0	0	0	1	0	0
SEQUENCE NUMBER								ACKNOWLEDGEMENT NUMBER							

5. Synchronization (SYN) segment : Segment used to establish connection.

<table>
<tr><td colspan="8"><_____8-bits_____></td><td colspan="8"><_____8-bits_____></td></tr>
<tr><td>1</td><td>ACK</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">SEQUENCE NUMBER</td><td colspan="8">ACKNOWLEDGEMENT NUMBER</td></tr>
<tr><td>1</td><td>CHK</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="8">MAX. NUMBER OF OUTSTANDING SEGMENTS</td></tr>
<tr><td colspan="16">RETRANSMISSION TIMEOUT VALUE IN MILLISECONDS</td></tr>
<tr><td colspan="16">CUMULATIVE ACK TIMEOUT VALUE IN MILLISECONDS</td></tr>
<tr><td colspan="16">NULL SEGMENT TIMEOUT VALUE IN MILLISECONDS</td></tr>
<tr><td colspan="8">MAX RETRANSMISSIONS</td><td colspan="8">MAX CUMULATIVE ACKS</td></tr>
<tr><td colspan="16">CONNECTION ID (32 BITS)</td></tr>
</table>

- MAX. NUMBER OF OUTSTANDING SEGMENTS : Maximum segments which can be sent in the network that haven't been ACKed yet.
- RETRANSMISSION TIMEOUT : The maximum time interval after which the client should retransmit a packet if it does not receive that packet's corresponding ACK.
- CUMULATIVE ACK TIMEOUT : The maximum time interval after which the server should retransmit an ACK segment if it does not receive another segment from the client.
- NULL SEGMENT TIMEOUT : The timeout value for sending NULL segment if the data segment has not been sent. (keep-alive mechanism)
- MAX RETRANSMISSIONS : The maximum number of time consecutive retransmissions will be sent before the connection is considered dead.
- MAX CUMULATIVE ACKS : The maximum number of acknowledgements that will be accumulated at the server side before sending an acknowledgment if a new segment is not received.
- CONNECTION ID : Each different connection will have a unique connection ID.

Features description

Connection setup :

Initially client and server start in LISTEN state.

1. Client chooses an initial sequence number x and sends a packet with SYN bit 1.
2. Server, on receiving a packet with SYN bit set, chooses an initial sequence number y and sends a packet to Client with SYN bit and ACK bit set with Sequence number y and ACKnum x+1 and enters SYN_RCVD state.
3. Client on receiving the packet in 2, sends a packet with ACK bit and DATA bit set, ACKnum as y+1 and data in payload section. Client enters ESTAB state and Server on receiving the packet enters ESTAB state.
   Connection is established.

For setting up the connection, the first data packet is sent from the sender to the receiver. The sequence number field is set to the pre-decided initial sequence number. A timer corresponding to this sequence number is set up to a pre-defined timeout value.

Data transmission :

The data to be sent is divided into segments.

The sequence number of a segment is calculated as Sequence number = (previous sequence number + 1) mod (window size).

If the sequence number is within the window size of the sender, the segment is prepared for transmission. The RUDP specific header fields are added to the segment and the segment is sent via the UDP transport layer protocol.

Sender :

1. State variables:
    1. send_base is the start index of the send window.
    2. nextseqnum is the sequence number to be used for the next outgoing packet.
2. 4 possible events :
    1. A packet arrives from the application layer
        1.1. When data is received from application, the sender checks the next available sequence number for the packet. If the sequence number is not within the sender's window, it is either buffered or returned to the upper layer for later transmission.
        1.2. Otherwise give the packet the sequence number nextseqnum and send it to the network layer. Save the packet in case it needs to be resent. Start timer nextseqnum.
        1.3. Increment nextseqnum and stop accepting input from the application layer if the window is full.
    2. Timer i fires:
        2.1. Send saved packet i to the network layer and restart timer i.
    3. An acknowledgment arrives that has a sequence number, i, in the window:
        3.1. Stop timer i so that it will not fire.
        3.2. Advance the window to begin at the first unacknowledged packet.
        3.3. Accept data from the application layer if the window has advanced.
    4. An acknowledgment arrives and its sequence number is not in the window:
        4.1. Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.

Receiver :

1. State variables
    1. rcv_base is the start index of the receive window.
2. 2 possible events :
    1. A packet arrives and its sequence number is in the window:
        1.1. Save it and send an acknowledgment for the packet to the network layer.
        1.2. Forward acknowledged packets at the beginning of the window to the application layer and advance rcv_base past these packets.
    2. A packet arrives and its sequence number is not in the window:
        2.1. Send an acknowledgment for the packet to the network layer. The sender must not have received the previous acknowledgment.

Connection close :

1. Client initiates closing of the connection by sending a packet with FIN bit set and sequence number at that time(say x). Client enters FIN_WAIT_1 state.
2. Server on receiving the previous packet, send a packet with ACK bit set and ACKnum at that time(say y). Server enters CLOSE_WAIT state.
3. Client on receiving, enters FIN_WAIT_2 state.
4. Server sends a packet with FIN bit set and sequence number y and enters LAST_ACK state.

5. On receiving, Client sends a packet with ACK bit set and ACKnum as y+1 and enters TIMED_WAIT state and after a timed wait, enters CLOSED state.
6. Server enters CLOSED state on receiving the packet.
   Connection is closed.

Timeouts :

A timer is set up for each segment while it is being transmitted. If the timer of a segment times out before an ACK is received for it, the segment is retransmitted and the timer is reset. If the ACK is received before the timeout of the retransmission, the timer is stopped immediately.

Scenarios and recovery mechanisms

If a SYN, DATA, NULL or FIN segment does not reach the server

- A segment does not reach the server in the case where it is dropped somewhere in the network layer or when the checksum does not match. In this case, the server will not send an ACK for the corresponding segment. Hence, the client will not receive the ACK within the retransmission timeout (specified in SYN segment). This will trigger the client to retransmit the packet.
- The client will repeat the above process N (N = Maximum retransmissions, specified in SYN segment) number of times until it does not receive the ACK from the server. After N times, the connection will be considered to be broken.

If an ACK does not reach the client

- When the client does not receive an ACK for a segment it sent within the cumulative ACK timeout (specified in SYN segment), it retransmits the segment. When the server receives the retransmitted packet, it will again send an ACK as long as the sequence number is between rcv_base-M and rcv_base-1 (M = Maximum number of outstanding segments). This continues until the client stops retransmitting the segment (ie.) when the client has successfully received the ACK for the segment.
- The server will keep accumulating a maximum of N (N = Maximum number of cumulative acknowledgements, specified in SYN segment) number of acknowledgements before sending an acknowledgement if it does not receive the next segment from the client. If this value is 0 then it indicates that an acknowledgment will be sent immediately after a data, null or close segment is received.

Broken connection

- The connection can break in case if the retransmission timer for a packet expires and the retransmission count exceeds the Maximum retransmission value. It can also break if the NULL segment timer on the server side expires.
- We can deal with this situation in a naive manner by setting up a new connection and send the SYN segment having the same attributes as the previous one. This new connection will retransmit all the packets from the beginning.