RUDP

The RUDP (Reliable User Datagram Protocol) API provided in the code facilitates reliable communication over UDP (User Datagram Protocol) connections.

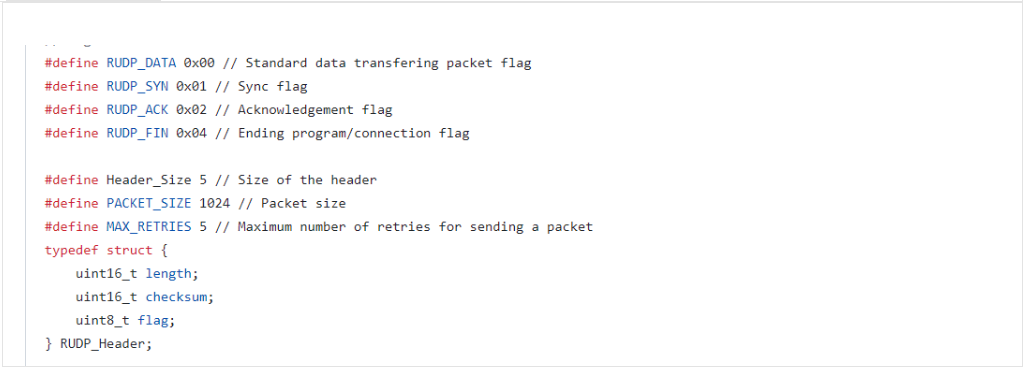
there are detailed overview of the API components:

Data Structures:

- **RUDP_Header**: Represents the header of an RUDP packet containing fields like length, checksum, and flag.

Flags:

- **RUDP_DATA**: Standard data transfer packet.
- **RUDP_SYN**: Synchronization for connection establishment.
- **RUDP_ACK**: Acknowledgment of received data.
- **RUDP_FIN**: Indicates the end of the program or connection.



```
#define RUDP_DATA 0x00 // Standard data transfering packet flag
#define RUDP_SYN 0x01 // Sync flag
#define RUDP_ACK 0x02 // Acknowledgement flag
#define RUDP_FIN 0x04 // Ending program/connection flag

#define Header_Size 5 // Size of the header
#define PACKET_SIZE 1024 // Packet size
#define MAX_RETRIES 5 // Maximum number of retries for sending a packet
typedef struct {
    uint16_t length;
    uint16_t checksum;
    uint8_t flag;
} RUDP_Header;
```

Functions:

- **calculate_checksum**: Calculates the one's complement checksum of binary data to ensure data integrity during transmission
- **generate_random_byte**: Generates a random byte used for handshake messages during connection establishment
- **rudp_close**: Closes RUDP connection associated with a socket file descriptor.
- **rudp_send**: When sending data over the RUDP (Reliable User Datagram Protocol) connection with a custom header, the sender includes essential information like the data payload, a checksum for error detection, a flag indicating the packet type, and the destination address. This custom header ensures reliable and efficient transmission of data between communicating nodes in the network.

- **rudp_recv**:
- receives data packets over the RUDP connection. It verifies packet headers, processes data, sends acknowledgments for received packets, handles timeouts, and manages errors to ensure reliable communication between sender and receiver nodes.

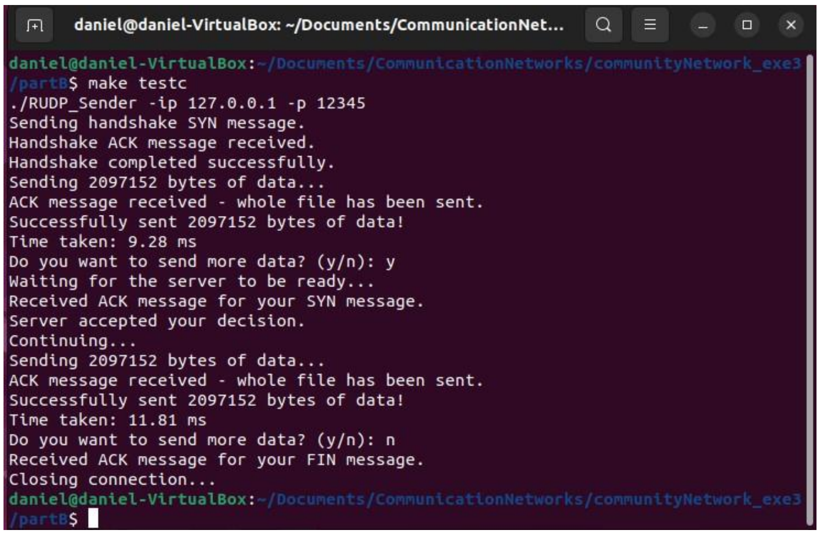
- **rudp_socket_receiver**: Sets up RUDP socket for receiver side, performs handshake with sender.
- **rudp_socket_sender**: Sets up RUDP socket for sender side, performs handshake with receiver.

Constants:

- **Header_Size**: Size of RUDP header in bytes.
- **PACKET_SIZE**: Maximum size of packet.
- **MAX_RETRIES**: Maximum retries for sending a packet.

Sender File:

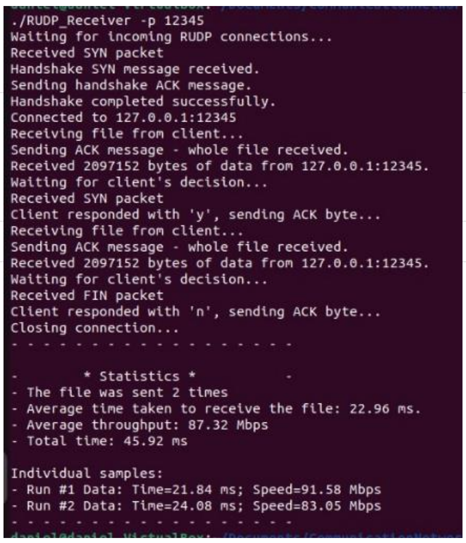
- Generates a random file of specified size.
- Creates RUDP socket and continuously sends data packets to receiver.
- Resends packets if acknowledgment not received within timeout.
- Allows user to decide whether to continue sending data or not.
- Sends termination signal (FIN) to receiver and closes connection.



```
daniel@daniel-VirtualBox:~/Documents/CommunicationNetworks/communityNetwork_exe3
/partB$ make testc
./RUDP_Sender -ip 127.0.0.1 -p 12345
Sending handshake SYN message.
Handshake ACK message received.
Handshake completed successfully.
Sending 2097152 bytes of data...
ACK message received - whole file has been sent.
Successfully sent 2097152 bytes of data!
Time taken: 9.28 ms
Do you want to send more data? (y/n): y
Waiting for the server to be ready...
Received ACK message for your SYN message.
Server accepted your decision.
Continuing...
Sending 2097152 bytes of data...
ACK message received - whole file has been sent.
Successfully sent 2097152 bytes of data!
Time taken: 11.81 ms
Do you want to send more data? (y/n): n
Received ACK message for your FIN message.
Closing connection...
daniel@daniel-VirtualBox:~/Documents/CommunicationNetworks/communityNetwork_exe3
/partB$
```

Receiver File:

- Waits for incoming connections from sender.
- Receives data packets, sends acknowledgment back to sender.
- Handles timeouts and errors during reception process.
- Waits for sender's decision to continue or terminate connection.
- Acknowledges termination and prints data transfer statistics.



```
./RUDP_Receiver -p 12345
Waiting for incoming RUDP connections...
Received SYN packet
Handshake SYN message received.
Sending handshake ACK message.
Handshake completed successfully.
Connected to 127.0.0.1:12345
Receiving file from client...
Sending ACK message - whole file received.
Received 2097152 bytes of data from 127.0.0.1:12345.
Waiting for client's decision...
Received SYN packet
Client responded with 'y', sending ACK byte...
Receiving file from client...
Sending ACK message - whole file received.
Received 2097152 bytes of data from 127.0.0.1:12345.
Waiting for client's decision...
Received FIN packet
Client responded with 'n', sending ACK byte...
Closing connection...
- - - - - - - - - - - - - - - - - -

-        * Statistics *          -
- The file was sent 2 times
- Average time taken to receive the file: 22.96 ms.
- Average throughput: 87.32 Mbps
- Total time: 45.92 ms

Individual samples:
- Run #1 Data: Time=21.84 ms; Speed=91.58 Mbps
- Run #2 Data: Time=24.08 ms; Speed=83.05 Mbps
- - - - - - - - - - - - - - - - - -
```

In summary, the sender sends data packets to the receiver, which acknowledges each received packet. The process continues until the sender decides to terminate the connection. The receiver continuously receives data packets, sends ACKs, and handles termination signals. Both sender and receiver ensure reliable data transfer over an unreliable network using the RUDP protocol.