are the sockets non blocking ?

The provided code snippet does not explicitly set the socket to non-blocking mode. By default, sockets are blocking. To make the socket non-blocking, you would need to use the `fcntl` function to set the `O_NONBLOCK` flag.

Here is how you can set the socket to non-blocking mode:

```
#include <fcntl.h>

// Set the socket to non-blocking mode
int flags = fcntl(sockfd, F_GETFL, 0);
fcntl(sockfd, F_SETFL, flags | O_NONBLOCK);
```

You should add this code after creating the socket and before using it in your `sendto` and `select` calls.

---

please do that uwu

To set the socket to non-blocking mode, you can use the `fcntl` function. Here is the modified code snippet with the necessary changes:

PROFI

```
#include <fcntl.h>

// Assuming sockfd is already created and initialized

// Set the socket to non-blocking mode
int flags = fcntl(sockfd, F_GETFL, 0);
fcntl(sockfd, F_SETFL, flags | O_NONBLOCK);

int total_chunks = (msg_len + CHUNK_SIZE - 1) / CHUNK_SIZE;

for (int i = 0; i < total_chunks; i++) {
    send_packet.seq_num = i + 1;
    send_packet.total_chunks = total_chunks;
    strncpy(send_packet.data, message + i * CHUNK_SIZE, CHUNK_SIZE);

    int attempts = 0;
    int ack_received = 0;

    while (!ack_received && attempts < 5) {
        sendto(sockfd, &send_packet, sizeof(Packet), 0,
               (const struct sockaddr *)addr, addr_len);
        printf("Sent chunk %d of %d\n", send_packet.seq_num,
send_packet.total_chunks);

        struct timeval tv;
        tv.tv_sec = 0;
```

```
        tv.tv_usec = TIMEOUT_MS * 1000;

        fd_set readfds;
        FD_ZERO(&readfds);
        FD_SET(sockfd, &readfds);

        int activity = select(sockfd + 1, &readfds, NULL, NULL, &tv);
        // Handle activity...
    }
}
```

This code sets the socket to non-blocking mode before entering the loop to send and receive packets.

---

do this to client too

To set the socket to non-blocking mode in the client as well, you can use the `fcntl` function. Here is an example of how you can modify the client code:

```
#include <fcntl.h>

// Assuming sockfd is already created and initialized

// Set the socket to non-blocking mode
int flags = fcntl(sockfd, F_GETFL, 0);
fcntl(sockfd, F_SETFL, flags | O_NONBLOCK);

int ack_received = 0;
int attempts = 0;

while (!ack_received && attempts < 5) {
    sendto(sockfd, &send_packet, sizeof(Packet), 0,
           (const struct sockaddr *)addr, addr_len);
    printf("Sent chunk %d of %d\n", send_packet.seq_num,
send_packet.total_chunks);

    struct timeval tv;
    tv.tv_sec = 0;
    tv.tv_usec = TIMEOUT_MS * 1000;

    fd_set readfds;
    FD_ZERO(&readfds);
    FD_SET(sockfd, &readfds);

    int activity = select(sockfd + 1, &readfds, NULL, NULL, &tv);

    if (activity > 0 && FD_ISSET(sockfd, &readfds)) {
        recvfrom(sockfd, &ack_packet, sizeof(Ack), 0, NULL, NULL);
        if (ack_packet.ack_num == send_packet.seq_num) {
            printf("Received ACK for chunk %d\n", ack_packet.ack_num);
            ack_received = 1;
```

PROFI

```
        }
    } else {
        attempts++;
        printf("No ACK received, retrying... (%d/5)\n", attempts);
    }
}
```

This code sets the socket to non-blocking mode before entering the loop to send and receive packets in the client as well.

---

to check if sockets is actually non blocking or not, skip acknowledgment for every 3 messages in server.c

To skip acknowledgment for every 3 messages in

and check if the socket is non-blocking, you can modify the code as follows:

1. Add a counter to keep track of the number of messages received.
2. Skip sending the acknowledgment for every 3rd message.

Here is the modified code:

```
#include <fcntl.h>

// Assuming sockfd is already created and initialized

// Set the socket to non-blocking mode
int flags = fcntl(sockfd, F_GETFL, 0);
fcntl(sockfd, F_SETFL, flags | O_NONBLOCK);

int message_count = 0;

while (1) {
    Packet recv_packet;
    Ack ack_packet;
    int n = recvfrom(sockfd, &recv_packet, sizeof(Packet), 0,
                     (struct sockaddr *)&client_addr, &client_len);

    if (n > 0) {
        printf("Received chunk %d of %d\n", recv_packet.seq_num,
recv_packet.total_chunks);

        message_count++;

        // Skip acknowledgment for every 3rd message
        if (message_count % 3 != 0) {
            ack_packet.ack_num = recv_packet.seq_num;
            sendto(sockfd, &ack_packet, sizeof(Ack), 0,
                   (const struct sockaddr *)&client_addr, client_len);
            printf("Sent ACK for chunk %d\n", ack_packet.ack_num);
```

```
        } else {
            printf("Skipping ACK for chunk %d\n", recv_packet.seq_num);
        }
    }
}
```

This code sets the socket to non-blocking mode and skips sending the acknowledgment for every 3rd message. This will help you verify if the socket is non-blocking by observing the behavior when acknowledgments are skipped.

---

to check if sockets is actually non blocking or not, skip acknowledgment for every 3 messages in server.c

To skip acknowledgment for every 3 messages in

and check if the socket is non-blocking, you can modify the code as follows:

1. Add a counter to keep track of the number of messages received.
2. Skip sending the acknowledgment for every 3rd message.

Here is the modified code:

```
#include <fcntl.h>

// Assuming sockfd is already created and initialized

// Set the socket to non-blocking mode
int flags = fcntl(sockfd, F_GETFL, 0);
fcntl(sockfd, F_SETFL, flags | O_NONBLOCK);

int message_count = 0;

while (1) {
    Packet recv_packet;
    Ack ack_packet;
    int n = recvfrom(sockfd, &recv_packet, sizeof(Packet), 0,
                     (struct sockaddr *)&client_addr, &client_len);

    if (n > 0) {
        printf("Received chunk %d of %d\n", recv_packet.seq_num,
recv_packet.total_chunks);

        message_count++;

        // Skip acknowledgment for every 3rd message
        if (message_count % 3 != 0) {
            ack_packet.ack_num = recv_packet.seq_num;
            sendto(sockfd, &ack_packet, sizeof(Ack), 0,
                   (const struct sockaddr *)&client_addr, client_len);
```

```
            printf("Sent ACK for chunk %d\n", ack_packet.ack_num);
        } else {
            printf("Skipping ACK for chunk %d\n", recv_packet.seq_num);
        }
    }
}
```

This code sets the socket to non-blocking mode and skips sending the acknowledgment for every 3rd message. This will help you verify if the socket is non-blocking by observing the behavior when acknowledgments are skipped.

**to check if sockets is actually non blocking or not, skip acknowledgment for every third messages while receive_message in server.c **

To skip acknowledgment for every third message while receiving messages in

, you can modify the `receive_message` function to include a counter that tracks the number of messages received and skips sending the acknowledgment for every third message.

Here is the modified code:

PROFI

```
#include <fcntl.h>

// Assuming sockfd is already created and initialized

// Set the socket to non-blocking mode
int flags = fcntl(sockfd, F_GETFL, 0);
fcntl(sockfd, F_SETFL, flags | O_NONBLOCK);

Packet received_packet;
Ack ack_packet;
char *received_data = NULL;
int total_chunks = 0;
int received_chunks = 0;
int message_count = 0;

while (received_chunks < total_chunks || total_chunks == 0) {
    int recv_len = recvfrom(sockfd, &received_packet, sizeof(Packet), 0,
                            (struct sockaddr *)addr, addr_len);

    if (recv_len > 0) {
        printf("Received chunk %d of %d\n", received_packet.seq_num,
received_packet.total_chunks);

        message_count++;

        // Skip acknowledgment for every 3rd message
        if (message_count % 3 != 0) {
            ack_packet.ack_num = received_packet.seq_num;
```

```
            sendto(sockfd, &ack_packet, sizeof(Ack), 0,
                   (const struct sockaddr *)addr, *addr_len);
            printf("Sent ACK for chunk %d\n", ack_packet.ack_num);
        } else {
            printf("Skipping ACK for chunk %d\n", received_packet.seq_num);
        }

        // Process received data
        if (received_data == NULL) {
            total_chunks = received_packet.total_chunks;
            received_data = calloc(total_chunks * CHUNK_SIZE + 1,
sizeof(char));
        }
        memcpy(received_data + (received_packet.seq_num - 1) * CHUNK_SIZE,
               received_packet.data, CHUNK_SIZE);
        received_chunks++;
    }
    // Small delay to prevent busy-waiting
    usleep(1000);
}
received_data[total_chunks * CHUNK_SIZE] = '\0';
return received_data;
```

This code sets the socket to non-blocking mode and skips sending the acknowledgment for every third message. This will help you verify if the socket is non-blocking by observing the behavior when acknowledgments are skipped.
