```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>

#define packetCount 10

typedef struct Packet {
   int id;
   int received;
} Packet;

Packet packets[packetCount];

void main() {
   for (int i = 0; i < packetCount; ++i) {
      packets[i].id = i + 1;
      packets[i].received = 0;
   }

   char *ip = "127.0.0.100";
   int port = 5567;
   srand(time(0));

   int sockfd;
   struct sockaddr_in server_addr, client_addr;
   char buffer[1024];
   socklen_t addr_size;
   int n;

   sockfd = socket(AF_INET, SOCK_DGRAM, 0);
   if (sockfd < 0) {
      perror("[-]Socket error");
      exit(1);
   }

   memset(&server_addr, '\0', sizeof(server_addr));
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(port);
   server_addr.sin_addr.s_addr = inet_addr(ip);

   n = bind(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr));
   if (n < 0) {
      perror("[+]Bind error");
      exit(1);
   }

   while (1) {
      bzero(buffer, 1024);
      addr_size = sizeof(client_addr);
      recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr *)&client_addr, &addr_size);
      sleep(1);

      int packet = atoi(buffer);
      if (packets[packet - 1].received == 1 || rand() % 5 == 0) {
         continue;
      } else if (rand() % 4 == 0) {
         printf("Server: Received corrupted packet %s. Sending negative acknowledgement\n", buffer);
         sprintf(buffer, "NACK %d", packet);
         sendto(sockfd, buffer, 1024, 0, (struct sockaddr *)&client_addr, sizeof(client_addr));
      } else {
         printf("Server: Received packet %s. Sending acknowledgement\n", buffer);
         sprintf(buffer, "ACK %d", packet);
         sendto(sockfd, buffer, 1024, 0, (struct sockaddr *)&client_addr, sizeof(client_addr));
         packets[packet - 1].received = 1;
      }
```

```
    }
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <pthread.h>

#define timeoutValue 5
#define packetCount 10
#define windowSize 3

typedef struct Packet {
    int id;
    int sent;
} Packet;

Packet packets[packetCount];
pthread_t thread[packetCount], receiveThread;
int sockfd;
struct sockaddr_in addr;
int windowStart, windowEnd;
int nextToSend;

void *sendPacket(void *sendPacket) {
    Packet *packet = (Packet *)sendPacket;
    char buffer[1024];

    while (packet->sent == 0) {
        printf("Client: Sending packet %d\n", packet->id);
        bzero(buffer, 1024);
        sprintf(buffer, "%d", packet->id);
        sendto(sockfd, buffer, 1024, 0, (struct sockaddr *)&addr, sizeof(addr));
        sleep(timeoutValue);
        if (packet->sent == 0)
            printf("Client: Timeout for packet %d\n", packet->id);
    }
}

void *receivePacket() {
    socklen_t addr_size = sizeof(addr);
    char buffer[1024];

    while (1) {
        bzero(buffer, 1024);
        recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr *)&addr, &addr_size);

        char msg[20];
        int packetID;
        sscanf(buffer, "%s%d", msg, &packetID);

        if (strcmp(msg, "NACK") == 0) {
            printf("Client: Received negative acknowledgment for packet %d\nSending again\n", packetID);
            pthread_cancel(thread[packetID - 1]);
            pthread_create(&thread[packetID - 1], NULL, sendPacket, (void *)&packets[packetID - 1]);
        } else if (strcmp(msg, "ACK") == 0) {
            printf("Client: Received acknowledgement for packet %d\n", packetID);
            packets[packetID - 1].sent = 1;

            if (windowStart == packetID - 1) {
                while (packets[windowStart].sent == 1) {
                    windowStart++;
                    if (windowEnd < packetCount)
                        windowEnd++;
                }
```

```
            }
        } else {
            printf("Client: Invalid message\n");
        }
    }
}

void main() {
    for (int i = 0; i < packetCount; ++i) {
        packets[i].id = i + 1;
        packets[i].sent = 0;
    }

    char *ip = "127.0.0.100";
    int port = 5567;
    char buffer[1024];
    socklen_t addr_size;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("[-]Socket error");
        exit(1);
    }

    memset(&addr, '\0', sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = inet_addr(ip);

    pthread_create(&receiveThread, NULL, receivePacket, NULL);

    windowStart = 0;
    windowEnd = windowStart + windowSize - 1;

     for (int i = windowStart; i <= windowEnd; ++i)
        pthread_create(&thread[i], NULL, sendPacket, (void *)&packets[i]);

    nextToSend = windowEnd + 1;

    while (windowStart != windowEnd) {
        if (nextToSend <= windowEnd && nextToSend < packetCount) {
            pthread_create(&thread[nextToSend], NULL, sendPacket, (void *)&packets[nextToSend]);
            nextToSend++;
        }
    }

    close(sockfd);
}

FTP server.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define PORT 1025
#define BUFFER_SIZE 1024

void handle_client(int client_sock);

int main() {
    int server_sock, client_sock;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_size;

    server_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (server_sock < 0) {
```

```
        perror("Socket error");
        exit(1);
    }

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("172.16.9.9");

    if (bind(server_sock, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        perror("Bind error");
        close(server_sock);
        exit(1);
    }

    if (listen(server_sock, 1024) < 0) {
        perror("Listen error");
        close(server_sock);
        exit(1);
    }

    printf("Server listening on port %d...\n", PORT);

    addr_size = sizeof(client_addr);
    client_sock = accept(server_sock, (struct sockaddr *) &client_addr, &addr_size);
    if (client_sock < 0) {
        perror("Accept error");
        close(server_sock);
        exit(1);
    }

    handle_client(client_sock);

    close(server_sock);
    return 0;
}

void handle_client(int client_sock) {
    char buffer[BUFFER_SIZE];
    char filename[100];
    FILE *fptr;

    while (1) {
        memset(buffer, 0, sizeof(buffer));

        ssize_t recv_len = recv(client_sock, buffer, sizeof(buffer), 0);
        if (recv_len <= 0) {
            if (recv_len == 0) {
                printf("Client disconnected\n");
            } else {
                perror("Recv error");
            }
            break;
        }

        if (strcmp(buffer, "PUT") == 0) {
            memset(buffer, 0, sizeof(buffer));

            recv(client_sock, buffer, sizeof(buffer), 0);
            strcpy(filename, buffer);
            memset(buffer, 0, sizeof(buffer));

            fptr = fopen(filename, "w");
            if (fptr == NULL) {
                perror("File open error");
                break;
            }

            while (1) {
                recv(client_sock, buffer, sizeof(buffer), 0);
```

```
            if (strcmp(buffer, "END$") == 0) {
                break;
            }
            fprintf(fptr, "%s", buffer);
            memset(buffer, 0, sizeof(buffer));
        }

        printf("File '%s' received successfully\n", filename);
        fclose(fptr);
    }

    else if (strcmp(buffer, "GET") == 0) {
        memset(buffer, 0, sizeof(buffer));

        sprintf(buffer, "%d", getpid());
        send(client_sock, buffer, sizeof(buffer), 0);
        memset(buffer, 0, sizeof(buffer));

        recv(client_sock, buffer, sizeof(buffer), 0);
        strcpy(filename, buffer);
        memset(buffer, 0, sizeof(buffer));

        fptr = fopen(filename, "r");
        if (!fptr) {
            printf("File '%s' does not exist\n", filename);
            strcpy(buffer, "404");
            send(client_sock, buffer, sizeof(buffer), 0);
        } else {
            strcpy(buffer, "200");
            send(client_sock, buffer, sizeof(buffer), 0);

            while (fgets(buffer, sizeof(buffer), fptr)) {
                send(client_sock, buffer, sizeof(buffer), 0);
                memset(buffer, 0, sizeof(buffer));
            }

            strcpy(buffer, "END$");
            send(client_sock, buffer, sizeof(buffer), 0);
            printf("File '%s' sent successfully\n", filename);
            fclose(fptr);
        }
    }

    else if (strcmp(buffer, "BYE") == 0) {
        close(client_sock);
        printf("Connection closed\n");
        break;
    }
  }
}
```

Ftp Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define PORT 1025
#define BUFFER_SIZE 1024

void handle_client(int client_sock);

int main() {
  int server_sock, client_sock;
  struct sockaddr_in server_addr, client_addr;
  socklen_t addr_size;
```

```
    server_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (server_sock < 0) {
        perror("Socket error");
        exit(1);
    }

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("172.16.9.9");

    if (bind(server_sock, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        perror("Bind error");
        close(server_sock);
        exit(1);
    }

    if (listen(server_sock, 1024) < 0) {
        perror("Listen error");
        close(server_sock);
        exit(1);
    }

    printf("Server listening on port %d...\n", PORT);

    addr_size = sizeof(client_addr);
    client_sock = accept(server_sock, (struct sockaddr *) &client_addr, &addr_size);
    if (client_sock < 0) {
        perror("Accept error");
        close(server_sock);
        exit(1);
    }

    handle_client(client_sock);

    close(server_sock);
    return 0;
}

void handle_client(int client_sock) {
    char buffer[BUFFER_SIZE];
    char filename[100];
    FILE *fptr;

    while (1) {
        memset(buffer, 0, sizeof(buffer));

        ssize_t recv_len = recv(client_sock, buffer, sizeof(buffer), 0);
        if (recv_len <= 0) {
            if (recv_len == 0) {
                printf("Client disconnected\n");
            } else {
                perror("Recv error");
            }
            break;
        }

        if (strcmp(buffer, "PUT") == 0) {
            memset(buffer, 0, sizeof(buffer));

            recv(client_sock, buffer, sizeof(buffer), 0);
            strcpy(filename, buffer);
            memset(buffer, 0, sizeof(buffer));

            fptr = fopen(filename, "w");
            if (fptr == NULL) {
                perror("File open error");
                break;
            }
```

```
            while (1) {
                recv(client_sock, buffer, sizeof(buffer), 0);
                if (strcmp(buffer, "END$") == 0) {
                    break;
                }
                fprintf(fptr, "%s", buffer);
                memset(buffer, 0, sizeof(buffer));
            }

            printf("File '%s' received successfully\n", filename);
            fclose(fptr);
        }

        else if (strcmp(buffer, "GET") == 0) {
            memset(buffer, 0, sizeof(buffer));

            sprintf(buffer, "%d", getpid());
            send(client_sock, buffer, sizeof(buffer), 0);
            memset(buffer, 0, sizeof(buffer));

            recv(client_sock, buffer, sizeof(buffer), 0);
            strcpy(filename, buffer);
            memset(buffer, 0, sizeof(buffer));

            fptr = fopen(filename, "r");
            if (!fptr) {
                printf("File '%s' does not exist\n", filename);
                strcpy(buffer, "404");
                send(client_sock, buffer, sizeof(buffer), 0);
            } else {
                strcpy(buffer, "200");
                send(client_sock, buffer, sizeof(buffer), 0);

                while (fgets(buffer, sizeof(buffer), fptr)) {
                    send(client_sock, buffer, sizeof(buffer), 0);
                    memset(buffer, 0, sizeof(buffer));
                }

                strcpy(buffer, "END$");
                send(client_sock, buffer, sizeof(buffer), 0);
                printf("File '%s' sent successfully\n", filename);
                fclose(fptr);
            }
        }

        else if (strcmp(buffer, "BYE") == 0) {
            close(client_sock);
            printf("Connection closed\n");
            break;
        }
    }
}
```