Sincronic

Tanase George

Facultatea de informatica, Universitatea Alexandru Ioan Cuza

1. Introducere

Scopul acestui proiect este de a crea un server TCP concurent care să demonstreze gestionarea conexiunilor concurente de la mai mulţi clienţi. Proiectul implică mai multe aspecte ale programării în reţea și a serverelor concurente, iar obiectivul principal este de a ilustra cum un server poate comunica cu mai mulţi clienţi în același timp și de a implementa o logică specifică în funcţie de mesajele primite de la acești clienţi.

2. Tehnologii Aplicate

TCP (Transmission Control Protocol) este un protocol de transport potrivit pentru acest tip de proiect deoarece asigura livrarea datelor fara erori si in ordinea corecta si gestioneaza controlul fluxului pentru a se asigura ca transmiterea datelor se face eficient. Aceste lucruri sunt importante deoarece aplicatia va dispune accesul a mai multor clienti simultan. Conceptele fundamentale ale TCP-ului sunt: socket-ul (mecanism bidirectional ce poate fi utilizat atat pentru a comunica intre procese de pe acelasi calculator, dar si pentru a asigura comunicarea in retea), adresa IP (un set de 4 numere separate prin punct), port (un numar pe 16 biti). Alte concepte folosite in proiect sunt: server-ul (ofera servicii clientilor), clientul (permite conectarea utilizatorului la server).

3. Structura Aplicatiei

1) Serverul - primeste conexiuni simultane de la mai multi clienti. Poate gestiona mai multe sesiuni de comunicare.

Conexiune Server:

```
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("Eroare la crearea socket-ului");
      exit(EXIT_FAILURE);
   }
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);
   if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
```

```
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0) {
        perror("Eroare");
        exit(EXIT_FAILURE);
 }
```

2) Clientul - trimite mesaje catre server

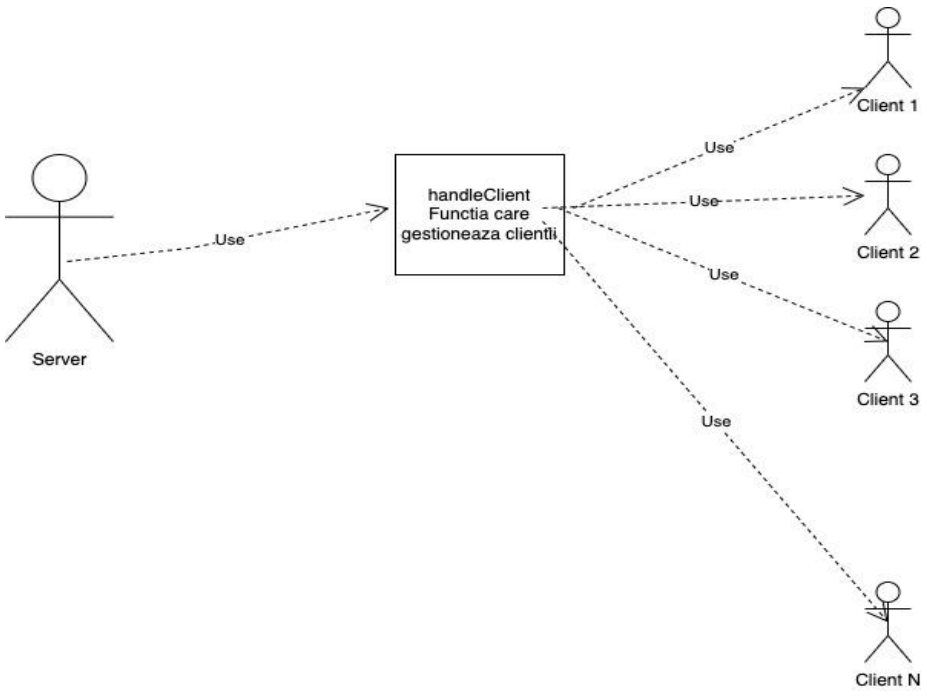
Conexiune client:

```
if ((clientSocket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Eroare la crearea socket-ului");
        exit(EXIT_FAILURE);
    }

    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP, &serverAddress.sin_addr) <= 0) {
        perror("Adresa serverului invalida");
        exit(EXIT_FAILURE);
    }
    if (connect(clientSocket, (struct sockaddr *)&serverAddress, sizeof(serverAddress))
< 0) {
        perror("Eroare la conectarea la server");
        exit(EXIT_FAILURE);
    }
```

Functia handleClient:

```
void handleClient(int clientSocket, int* clients, int* numReceivedMessages, char
mesaje[MAX_CLIENTS][1024]) {
    char buffer[1024] = {0};
    int bytesReceived;
    int M = 2;
    memset(buffer, 0, sizeof(buffer));
    bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);
    if (bytesReceived <= 0) {
        close(clientSocket);
        return;
    }
    strncpy(mesaje[*numReceivedMessages], buffer, 1023);
    mesaje[*numReceivedMessages][1023]='\0';
    clients[*numReceivedMessages] = clientSocket;
    (*numReceivedMessages)++;
    if (*numReceivedMessages >= MIN) {
        int messagesMatch = 1;
```

```
            for(int i=0; i < *numReceivedMessages-1; i++){
               for(int j=0; j<strlen(mesaje[i]); j++){
                  if(mesaje[i][j]!=mesaje[i+1][j]){
                     messagesMatch=0;
                    break;
                  }
               }
               if(!messagesMatch)
                  break;
            }
            if (messagesMatch) {
               char *response = "continua";
               for (int i = 0; i < *numReceivedMessages; i++) {
                  send(clients[i], response, strlen(response), 0);
               }
            } else {
               char *response = "gata";
               for (int i = 0; i < *numReceivedMessages; i++) {
                  send(clients[i], response, strlen(response), 0);
                  close(clients[i]);
               }
            }

         *numReceivedMessages = 0;
      }
   }
```



4. Aspecte de implementare

Un scenariu real de utilizare ar putea fi dezvoltarea de chat-uri sau platforme de comunicare in timp real.

Altul ar putea fi: serverul poate primi date de la diferite dispozitive si poate raspunde la comenzi de la utilizatori.

5. Concluzii

În concluzie, proiectul descrie dezvoltarea unei aplicaţii server-client pentru gestionarea comunicării în timp real între mai mulţi clienţi și server. Serverul poate primi mesaje de la clienţi și poate răspunde în funcţie de logica specifică a aplicaţiei. Proiectul are potenţialul de a fi folosit într-o varietate de scenarii, cum ar fi chat-uri, jocuri online, aplicaţii colaborative și multe altele.

Bibliografie

https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php