RAPORT TEHNIC al proiectului SINCRON

Grigore D Valerian - grupa 2A3

6 Decembrie 2022

1 Introducere

Proiectul **Sincron** presupune crearea unui server TCP concurent la care se pot conecta simultan maxim N clienti. Serverul primeste mesaje, din S in S minute, de la macar M dintre clienti, unde M este strict mai mic decat N. Daca mesajele primite de la cei M clienti nu coincid, atunci serverul va trimite celor N clienti mesajul "gata", dupa care deconecteaza toti clientii. Daca mesajele coincid, serverul va trimite tuturor clientilor mesajul "continua" si va astepta urmatoarele mesaje de la alti doi clienti ai sai.

2 Tehnologii Utilizate

Pentru comunicarea clienti-server vom folosi protocolul **TCP**. Un protocol, in contextul retelelor de calcuclatoare, este un set de reguli si proceduri care guverneaza modul in care datele sunt transmise. Scopul protocolului TCP este acela de a controla transferul de date in asa fel incat acesta sa fie de incredere.

Doua dintre caracteristicile protocolului TCP:

- Toate pachetele ajung la destinatie; niciun pachet nu este pierdut.
- Toate pachetele sunt reasamblate in ordine.

Asadar, in cadrul proiectului **Sincron**, mesajele transmise de cei M clienti trebuie sa ajunga la server exact asa cum au fost trimise pentru a verifica daca acestea coincid, mai departe serverul trimitand comenzile respective ("gata" sau "continua") care trebuie sa ajunga in siguranta la cei N clienti.

3 Arhitectura aplicatiei

Pe rand, clientii se vor conecta la server prin adresa IP si PORT-ul stabilite de server. Dupa conectare, clientii vor transmite date(in cazul nostru mesaje text) catre server, acesta fiind responsabil cu primirea, verificarea datelor si executarea anumitor comenzi dupa caz. Acesta are dreptul de a deconecta toti clientii comunicandu-le un anume "avertisment" inainte, sau de a-i pastra conectati.

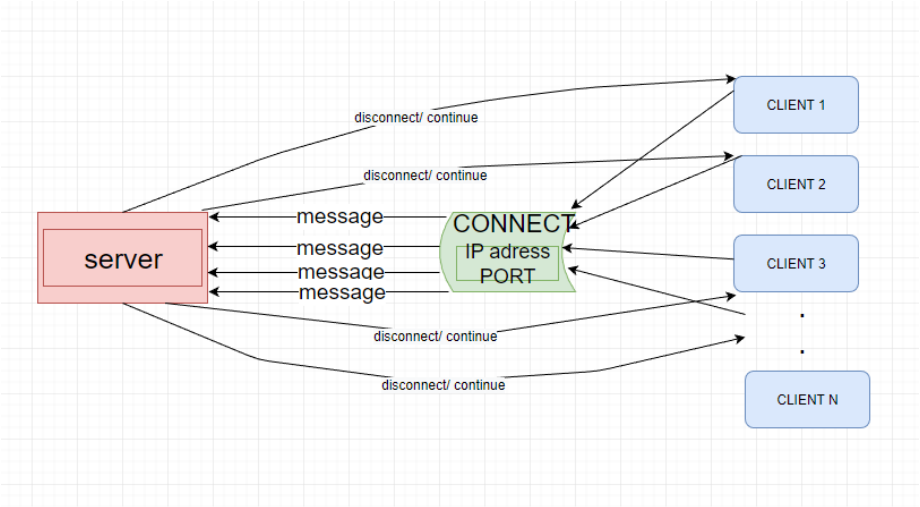

Figure 1: Sincron Diagram

4 Detalii de implementare

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

/*portul folosit*/
#define PORT 4444

int main()
{

    int sockfd; // descriptor socket
    int ret;
    struct sockaddr_in server; // structurile pt server si clienti

    int newSocket;
    struct sockaddr_in from;

    socklen_t addr_size;
    char buffer[1024];
    pid_t childpid;

    /*creare socket*/
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        printf("[-]Error in connection.\n");
        exit(1);
    }
    printf("[+]Server Socket is created.\n");

    /* umplem structura folosita de server */
    /*stabilirea familiei de socket-uri*/
    memset(&server, '\0', sizeof(server));
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
43  atasam socketul */
44   = bind(sockfd, (struct sockaddr *)&server, sizeof(server));
45  (ret < 0)
46
47   printf("[-]Error in binding.\n");
48   exit(1);
49
50  ntf("[+]Bind to port %d\n", 4444);
51
52  punem serverul sa asculte daca vin clienti sa se conecteze */
53  listen(sockfd, 10) == 0
54
55   printf("[+]Listening....\n");
56
57  e
58
59   printf("[-]Error in binding.\n");
60
61
62  servim in mod concurent clientii... */
63  le (1)
64
65
66   /* a venit un client, acceptam conexiunea */
67   newSocket = accept(sockfd, (struct sockaddr *)&from, &addr_size);
68   /* eroare la acceptarea conexiunii de la un client */
69
70   if (newSocket < 0)
71   {
72       exit(1);
73   }
74   /*s-a reusit conectarea si se va afisa mesajul respectiv*/
75   printf("Connection accepted from %s:%d\n", inet_ntoa(from.sin_addr), ntohs(from.sin_port));
76
```

```
61
62      /* servim in mod concurent clientii... */
63      while (1)
64      {
65
66          /* a venit un client, acceptam conexiunea */
67          newSocket = accept(sockfd, (struct sockaddr *)&from, &addr_size);
68          /* eroare la acceptarea conexiunii de la un client */
69
70          if (newSocket < 0)
71          {
72              exit(1);
73          }
74          /*s-a reusit conectarea si se va afisa mesajul respectiv*/
75          printf("Connection accepted from %s:%d\n", inet_ntoa(from.sin_addr), ntohs(from.sin_port));
76
77          if ((childpid = fork()) == 0)
78          {
79              close(sockfd);
80
81              while (1)
82              {
83                  /*daca comanda trimisa de client e ":exit" inseamna ca acesta se va deconecta de pe serve
84                  recv(newSocket, buffer, 1024, 0);
85                  if (strcmp(buffer, ":exit") == 0)
86                  {
87                      printf("Disconnected from %s:%d\n", inet_ntoa(from.sin_addr), ntohs(from.sin_port));
88                      break;
89                  }
90                  /*clientul trimite mesaje catre server*/
91                  else
92                  {
93                      printf("Client[%d]: %s\n", ntohs(from.sin_port), buffer);
94                      send(newSocket, buffer, strlen(buffer), 0);
95                      bzero(buffer, sizeof(buffer));
96                  }
97              }
98          }
99      }
00
01      close(newSocket);
02
03      return 0;
04  }
```

5 Concluzii

Aplicatia nu este finalizata inca. Este nevoie de verificarea tuturor mesajelor pe care cei M clienti le trimit serverului pentru a vedea daca acestea coincid, pentru a decide daca clientii vor fi deconectati de la server sau daca vor putea transmite in continuare mesaje.

6 Bibliografie

References

- https://profs.info.uaic.ro/ computernetworks/files/NetEx/S9/servTcpCSel.c
- https://profs.info.uaic.ro/ computernetworks/files/NetEx/S9/cliTcp.c
- https://profs.info.uaic.ro/ computernetworks/files/NetEx/S9/Makefile
- https://profs.info.uaic.ro/ ioana.bogdan/