GridAdmin

Munteanu Denisa, grupa B4

Facultatea de Informatică, Universitatea Alexandru Ioan Cuza, Iași
denisa.munteanu@info.uaic.com

Abstract. Proiectul constă în realizarea unei aplicații client-server care administrează o rețea de calculatoare a căror adresă IP este aflată în fișierul text **ip_addrs.txt**. Cu ajutorul acestei aplicații, unicul administrator poate interoga starea pcurilor, le poate închide și poate efectua comenzi bash.

Keywords: rețea de calculatoare, client-server, TCP

1 Introducere

O rețea de calculatoare reprezintă un ansamblu de calculatoare interconectate prin intermediul unor medii de comunicație în scopul utilizării în comun de către un număr foarte mare de utilizatori a tuturor resurselor fizice(hardware), logice(software de bază și aplicații) și informaționale (baze de date) associate calculatoarelor din rețea. Prin astfel de rețele de calculatoare se poate asigura o integrare informatică a unui număr foarte mare de utilizatori la nivel local, regional și mondial. Programarea în rețea implică trimiterea de mesaje și date între aplicații ce rulează pe pc-uri aflate într-o rețea locală sau conectată la Internet. Aplicația prezentată gestionează o rețea de calculatoare, îi permite administratorului să vizualizeze și să modifice stările pc-urilor, respectiv să execute comenzi bash pe fiecare calculator.

2 Tehnologii utilizate

2.1 TCP

TCP (Transimssion Control Protocol) este un protocol de transport orientat conexiune, oferă calitate maximă serviciilor, fără pierdere de informații, integrand mecanisme de stabilire și eliberare a conexiunii. Aceste conexiuni se identifică în mod unic prin perechi reprezentate de adresa ip:port. Adresa IP este o etichetă asignată unei entități întro rețea de calculatoare în timp ce portul identifică procesele ce rulează pe un sistem de calcul.

Serverul și clientul iau parte deopotrivă la realizarea conexiunii. Serverul oferă o deschidere pasivă, așteptând apariția unei cereri din partea clientului care realizează o deschidere active. Inițializarea conexiunii are loc prin parcurgerea unor pași specifici, metodă cunoscută sub numele de „three-way handshaking”.

Am ales să folosesc TCP, deoarece acest protocol facilitează transmisia de date sigură și nu permite pierderea acestora față de UDP care asigura o viteză mai mare, dar nu garantează acuratețea informațiilor.

2.2 SQLite

SQLite este o bibliotecă din limbajul C care implementează o bază de date SQL rapidă, de sine stătătoare, fiabilă, cu toate funcționalitățile din SQL. Este cel mai folosit database engine datorită rapidității și simplității de folosire. Avantajul folosirii SQLite este că nu e necesar un alt server, precum în cazul Oracle, iar toate datele sunt stocate doar într-un fișier pe disc, de unde rezultă rapiditatea față de bazele de date de tip client-server.

Serverul apelează la o bază de date relațională prin intermediul bibliotecii SQLite pentru a stoca utilizatorul și starea fiecărui PC cu adresa IP precizată.

2.3 Posix threads

Un thread (fir de execuție) este folosit pentru a eficientiza execuția programelor. Spre deosebire de procese, thread-urile partajează spațiul de adrese al procesului de care aparțin, folosirea lor are o serie de avantaje:

- Crearea/distrugerea unui fir de execuție durează mai puțin decât crearea/distrugerea unui process
- Durata context-switch-ului între firele de execuție ale aceluiași process este foarte mica, nefiind necesară comutarea spațiului de adrese
- Comunicarea între firele de execuție este realizată prin modificarea unor zone de memorie din spațiul comun de adrese

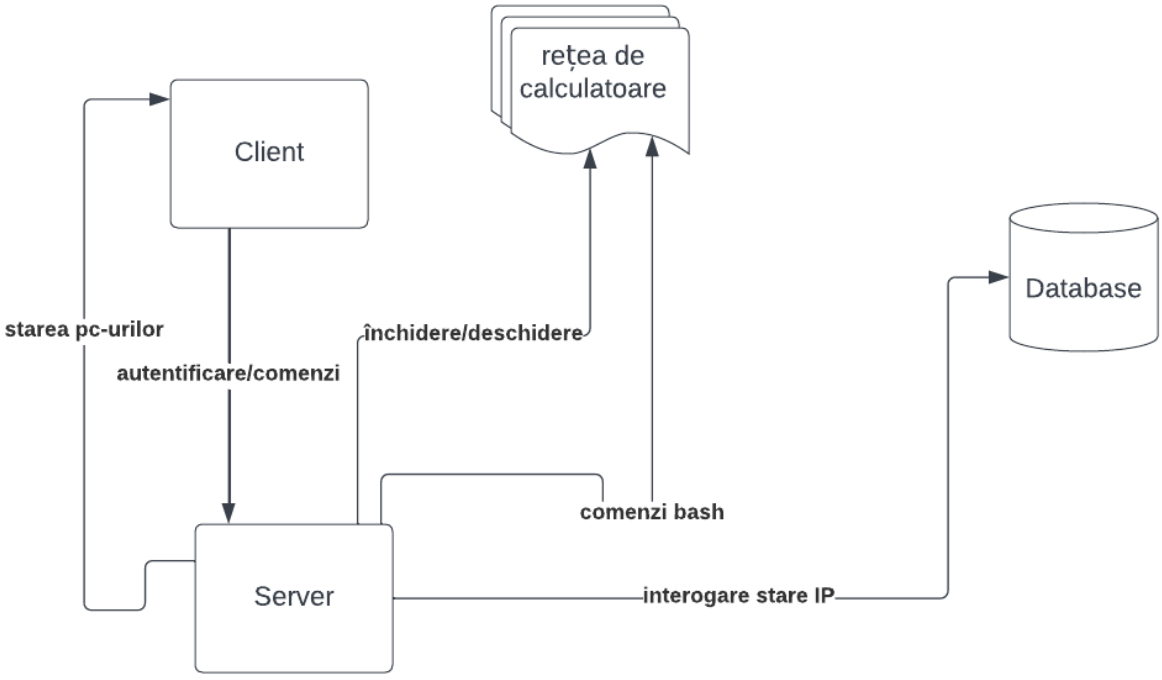
În cazul aplicației GridAdmin, se folosesc thread-uri pentru a putea deserve clienții într-un mod concurrent și rapid.

3 Arhitectura aplicației

3.1 Concepte aplicate

- Serverul va fi implementat folosind TCP.
- Comunicarea se va realiza prin intermediul socket-urilor utilizând apeluri read/write.
- Serverului i se atribuie un socket, după așteaptă administratorul să se conecteze. Îi validează datele de conectare și așteaptă comenzi de la acesta.
- Serverul este responsabil și cu accesul la baza de date stocate local. Clientul nu va avea posibilitatea de a trimite efectiv interogări SQL la baza de date, ci va delega serverul din rațiuni de Securitate și acces controlat la informații. Sunt stocate local detalii despre fiecare PC conectat: nume de utilizator, IP, stare (închis, deschis).
- Serverul va folosi conexiunea prin ssh pentru a efectua comenzi pe calculatoarele din rețea, precum shutdown pentru a închide PC-urile și alte comenzi bash.

3.2 Diagrama aplicației



4 Detalii de implementare

- Serverul va fi implementat folosind TCP.
- Comunicarea se va realiza prin intermediul socket-urilor utilizând apeluri read/write.

```
   /* crearea unui socket */
  if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
    {
      perror ("[server]Eroare la socket().\n");
return errno;
    }

/* atasam socketul */
  if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) ==
-1)
    {
      perror ("[server]Eroare la bind().\n");
return errno;
    }



    /* citim date primite  */
       int len;
        if (read (tdL.cl, &len , sizeof(int)) <= 0)
           {
             printf("[Thread %d]\n",tdL.idThread);
             perror ("Eroare la read() de la client.\n");

           }
        char user[len];
    if (read (tdL.cl, user, len) <= 0)
           {
             printf("[Thread %d]\n",tdL.idThread);
             perror ("Eroare la read() de la client.\n");

           }

       /* returnam mesajul clientului */
 if (write (tdL.cl, &ans, sizeof(int)) <= 0)
 {
  printf("[Thread %d] ",tdL.idThread);
  perror ("[Thread]Eroare la write() catre client.\n");
 }
else
 printf ("[Thread %d]Mesajul a fost trasmis cu succes.\n",tdL.idThread);
```

- Serverului i se atribuie un socket, după așteaptă administratorul să se conecteze. Îi validează datele de

conectare și așteaptă comenzi de la acesta. Nu am stocat datele de conectare într-o bază de date, deoarece erau unice.

```
/* validarea datelor de conectare*/
if(strcmp(user, "admin")==0 && strcmp(passw, "adminpass")==0) ans=1;
```

- Serverul este responsabil și cu accesul la baza de date stocate local. Clientul nu va avea posibilitatea de a trimite efectiv interogări SQL la baza de date, ci va delega serverul din rațiuni de Securitate și acces controlat la informații. Sunt stocate local datele de conectare ale administratorului și detalii despre fiecare PC conectat: IP, stare (închis, deschis).

```
sqlite3 *db;
char *err_msg = 0;
sqlite3_stmt* stmt;

/* deschiderea bazei de date*/
if (sqlite3_open ("connection.db", &db) != SQLITE_OK) {
    printf("Error opening database.\n");
    return ;
}

/* preluarea informaţiilor din baza de date */
char *sql = "SELECT * FROM Conn;";

int rc= sqlite3_prepare_v2(db, sql, -1, &stmt, 0);
if(rc!=SQLITE_OK){
    printf("Error fetching data");
    sqlite3_close(db);
}

/* stocarea ip-ului și a stării lui */
char row[1000];
while(sqlite3_step(stmt)==SQLITE_ROW){

    sprintf(row,"%s    %s\n",    sqlite3_column_text(stmt,0),
    sqlite3_column_text(stmt,1));
    strcat(message, row);
}
mlen=strlen(message);
sqlite3_finalize(stmt);
sqlite3_close(db);
```

- Serverul actualizează starea calculatoarelor în baza de date.

```
void Update_cts(){

    /*deschidem baza de date*/
    sqlite3 *db;
```

```
    char *err_msg = 0;

    int rc = sqlite3_open("connection.db", &db);

    if (rc != SQLITE_OK) {

        fprintf(stderr,"Cannot open database: %s\n", sqlite3_errmsg(db));
        sqlite3_close(db);

        return ;
    }


    /* actualizăm starea pentru fiecare IP din fișier */

    FILE *fp=fopen("ip_addrs.txt","r");
    if(fp==NULL){
    perror("Unable to open file!");
    exit(1);
    }
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    char ip[128];
    while(fgets(ip, sizeof(ip),fp)!=NULL){

        ip[strlen(ip)-1]='\0';
        char insert_ip[1000];
        char sir[1000];
        char state[5];
        struct sockaddr_in fileip;
        fileip.sin_family=AF_INET;

      /* trebuie folosit alt port decât cel folosit de server */
        fileip.sin_port=htons(4496);

        inet_pton(AF_INET, ip, &fileip.sin_addr);
        sprintf(insert_ip, "UPDATE Conn SET State = 'off' WHERE IP =
'%s';", ip);
        rc=sqlite3_exec(db, insert_ip,0,0,&err_msg);
        strcpy(state,"on");
        if (connect(sockfd, (struct sockaddr *) &fileip, sizeof(fileip))
== -1)
        {
            /*eroarea 113 indică faptul că PC-ul este pornit, dar refuză
            conexiunea*/
            if(errno==113)strcpy(state,"off");

        }
        //else printf("%s %s\n", ip, state);
        sprintf(insert_ip, "UPDATE Conn SET State = '%s' WHERE IP = '%s';",
state, ip);
```

```
        rc=sqlite3_exec(db, insert_ip,0,0,&err_msg);

    }
    fclose(fp);
    if (rc != SQLITE_OK ) {

        fprintf(stderr, "SQL error: %s\n", err_msg);

        sqlite3_free(err_msg);
        sqlite3_close(db);

        return ;
    }
    sqlite3_close(db);

}
```

- Serverul va folosi conexiunea prin ssh pentru a efectua comenzi pe calculatoarele din rețea, precum shutdown pentru a închide PC-urile și alte comenzi bash.

```
/* închiderea PC-ului ales */
char syscom[1000];
sprintf(syscom, "ssh -t %s@%s 'sudo shutdown -h 0'", chosen_user, IP2);
printf("\n\n%s\n\n", syscom);
system(syscom);



/*partea de executare a comenzilor bash*/
FILE *com;
char path[1035], output[100000]="";

char INPUT[10000];
sprintf(INPUT,"ssh -t %s@%s '%s' ",chosen_user,IP2,input);
printf("\n%s\n", INPUT);

/* Executăm comanda dată de client */
com = popen(INPUT, "r");
if (com == NULL)
{
    printf("Failed to run command\n" );
    exit(1);
 }

 /* Citim output-ul linie cu linie și îl concatenăm în string-ul
      Care va fi trimis clientului */
  while (fgets(path, sizeof(path), com) != NULL) {

        strcat(output, path);
  }
  int len=strlen(output);
```

```
//printf("Lungime %d\n Output: %s", len, output);

 /* close */
 pclose(com);
```

5 Concluzii

- Proiectul nu este 100% efficient, nu am luat in considerare toate cazurile posibile de erori.
- Codul mai poate fi modularizat pentru a fi mai ușor de citit.
- Ar putea fi adăugate mai multe funcții pentru gestionarea calculatoarelor.
- Nu am putut adăuga funcția wake-on-lan, deoarece am implementat serverul și clientul pe mașini virtuale.

Bibliografie

1. Author, F.: Article title. Journal 2(5), 99–110 (2016).
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016).
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999).
4. Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010).
5. LNCS Homepage, http://www.springer.com/lncs, last accessed 2022/12/02.
6. Diagrama aplicației, https://www.lucidchart.com/pages/
7. Codul de la care am început serverul, https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
8. Informații tehnologii utilizate, https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
9. Codul de la care am început clientul, https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
10. Interogare stare IP, https://stackoverflow.com/questions/9498831/how-to-check-if-an-ip-address-is-reachable-in-c-for-linux
11. Comanda popen pentru efectuarea comenzilor, https://stackoverflow.com/questions/646241/c-run-a-system-command-and-get-output
12. Implementarea comenzii SELECT * FROM, https://www.includehelp.com/c-programs/get-records-from-a-database-table-in-sqlite.aspx
13. Implementarea unor comenzi Sqlite in C, https://zetcode.com/db/sqlitec/