MyFileTransferProtocol

Tugui Ioana Madalina

Facultatea de Informatica anul 2 grupa B2

1 Introducere

Aplicatia va fi una facuta pentru transmiterea de fisiere intre client si server. Serverul va fi cel care va stoca intr-un director specific userilor autentificati(ce au permisiuni pe server de logare) toate fisierele pe care user-ul doreste sa le salveze.
Clientul va putea beneficia de anumite comenzi: nano, delete (pentru fisier), mkdir, rmdir, uploadFolder,uploadFile, get.

2 Tehnologii utilizate

Pentru acest proiect se va utiliza modelul TCP concurent.
Am ales sa utilizez TCP concurent pentru a putea avea un numar nelimitat de clienti conectati la server.
Concurenta se va realiza prin crearea unui proces copil pentru fiecare client, care va executa toate functiile necesare pentru a transmite catre client inapoi un raspuns valid si asteptat.
Un alt avantaj pentru modelul TCP este acela de siguranta in ceea ce priveste primirea datelor inapoi la client si acceptarea asigurata a clientului.
Protocolul utilizat pentru transmiterea fisierelor va fi cel FTP.

3 Arhitectura aplicatiei

Mai jos voi descrie schematic modelul TCP si cel FTP, dar si o introducere in functiile pe care le voi utiliza.

Fig. 1. TCP MODEL

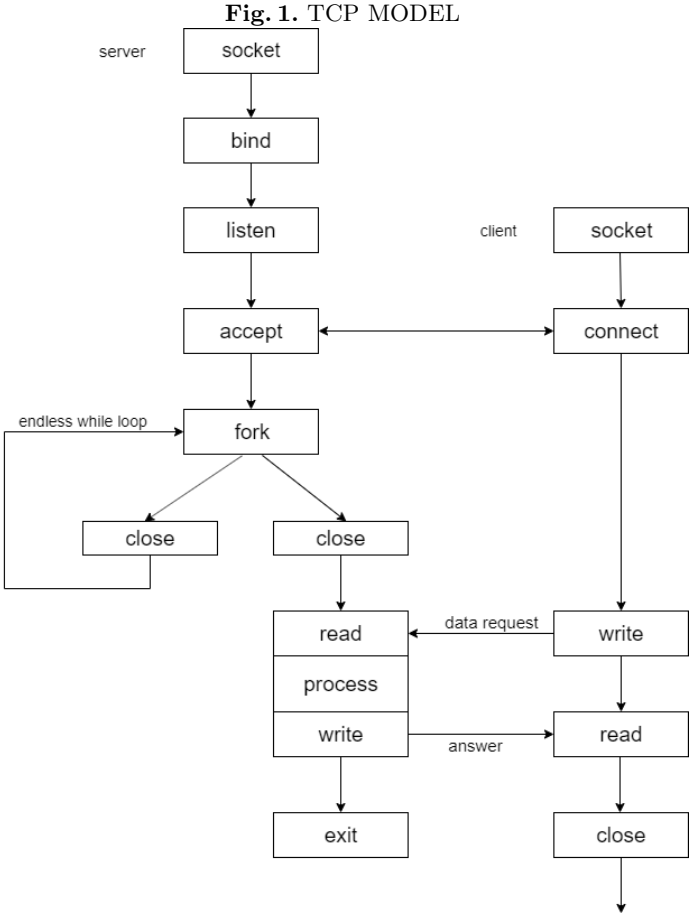


Fig. 2. FTP MODELL

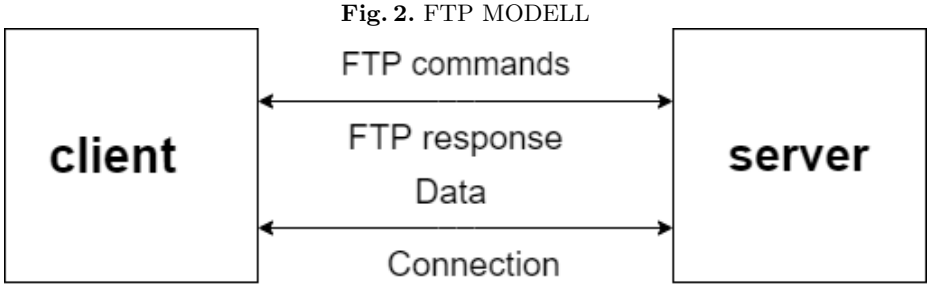


Fig. 3. PROJECT MODEL

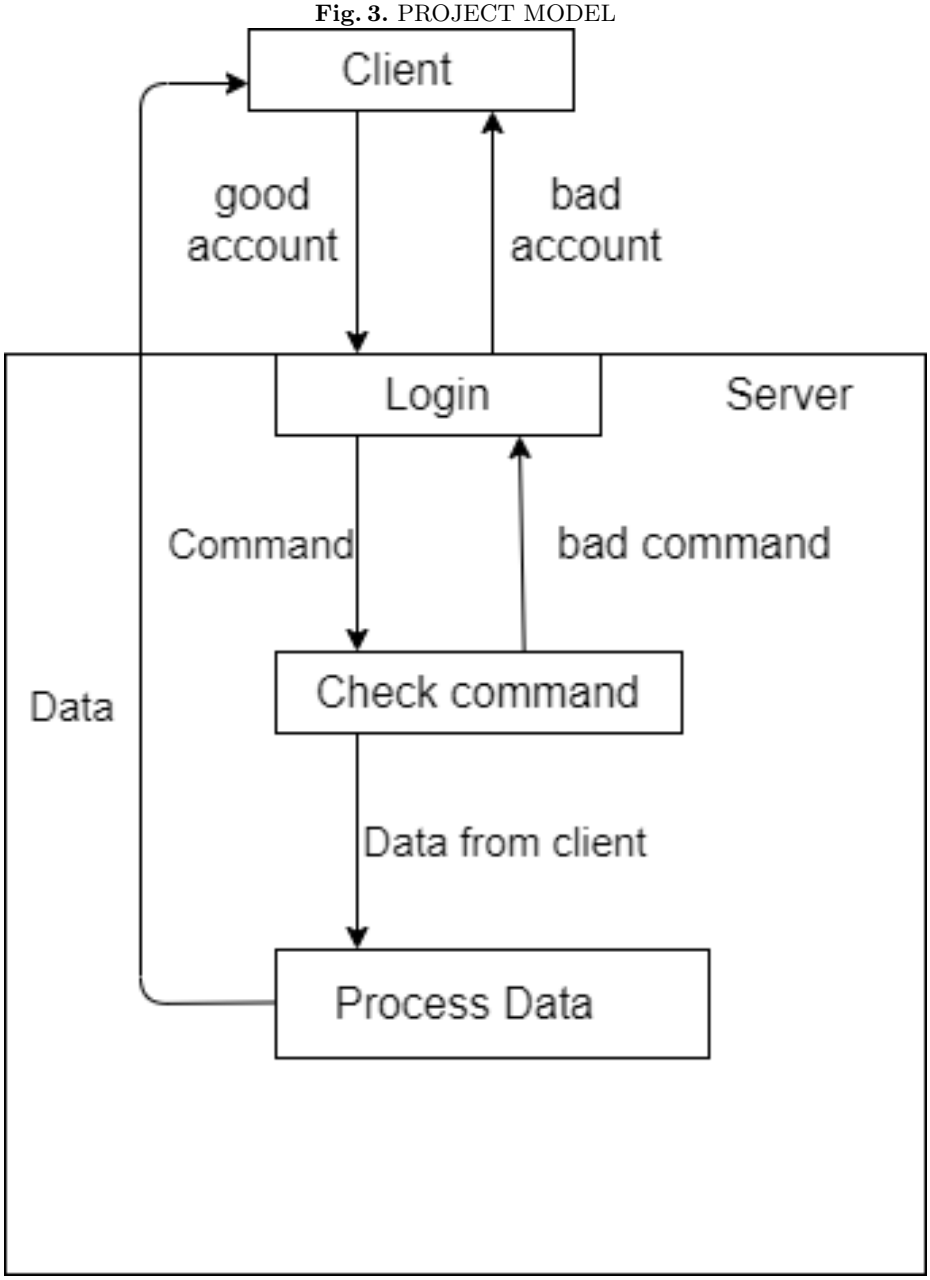
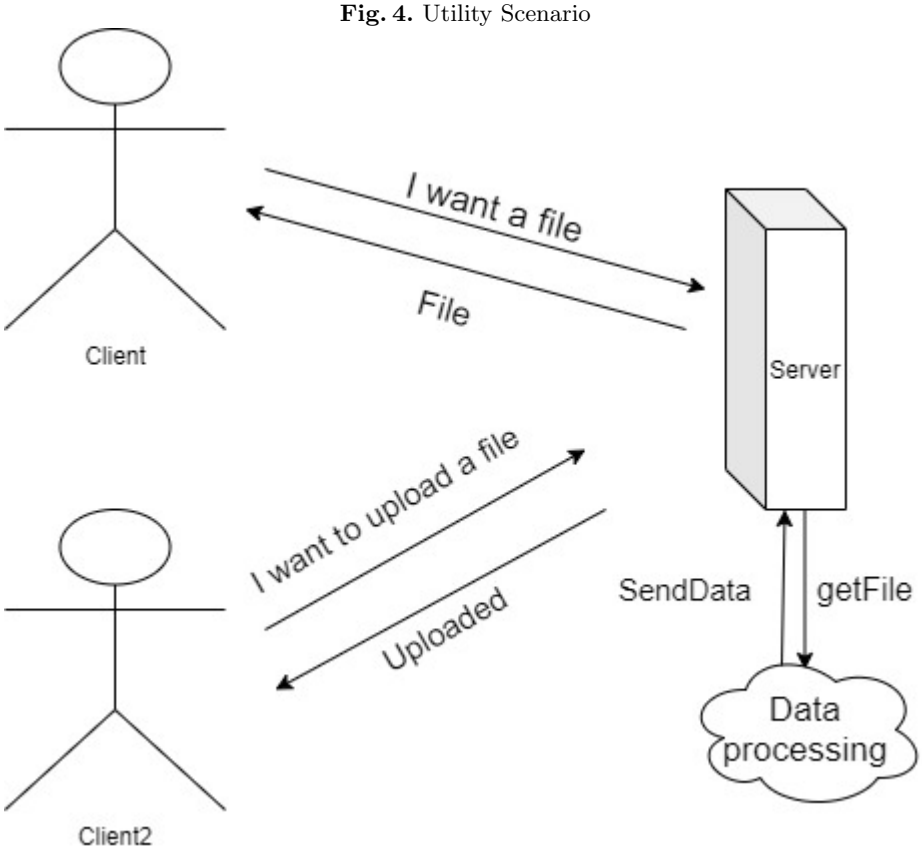


Fig. 4. Utility Scenario



4 Detalii de implementare

Proiectul va fi implementat in C. Mai jos sunt detaliate anumite functii pe care se va baza acesta.

Functia de downloadsFile() din client este cea care ajuta la descarcarea unui fisier primit de la server.

```
void downloadFile(char path[], char fileInfo[], int type,int sd,char* fileTo
{
  chdir(path);
  if (read (sd, fileInfo, 9999) < 0)
  {
    perror ("[client]Eroare la read() de la server.\n");
    return errno;
  }
  if(strcmp(fileInfo,"File/Folder does not exist!~~")==0)
  {
    printf("File/Folder does not exist!\n");
    return;
  }
  else if(type==1)
  {
    printf("Downloading..\n");
    struct stat path_status;
    int* size = NULL;
    char* fileData=(char*)malloc(9999);
    getInfo(&fileData,&size,fileInfo);
    FILE* fd;
    fd = fopen(path,"wb");
    if(fd==-1)
    {
      printf("Can't open file!\n");
      return;
    }
    fwrite(fileData,size,1,fd);
    fclose(fd);
    printf("Download of a file complete!\n");
  }
  else if(type==2)
  {
    mkdir(fileToGet,0777);
```

```
    printf("Downloading..\n");
    while(strcmp(fileInfo,"DoneTransfering")!=0)
    {
      printf("File␣%s\n",fileInfo);
      if(strstr(fileInfo,".")==0)
      {
          mkdir(fileInfo,0777);
      }
      else
      {
        FILE* fd;
        fd=fopen(fileInfo,"w");
        close(fd);
      }
      if (read (sd, fileInfo, 9999) < 0)
      {
        perror ("[client]Eroare␣la␣read()␣de␣la␣server.\n");
        return errno;
      }
    }
    printf("Download␣of␣a␣directory␣complete!\n");
    chdir("..");
  }
}
```

Functia de mai jos este una folosita pentru autentificarea unui client. Un client daca nu este autentificat, atunci nu va putea executa comenzi. Pe server se afla un folder "AccesAutentificare" in care sunt inregistrati toti utilizatorii server-ului care au, sau nu au voie sa execute comenzi(WhiteList/BlackList).

```
bool check(char* password, char* username, char* file)
{
        char path[100];
        strcpy(path,"ClientiServer/AccesAutentificare/");
        strcat(path,file);
        FILE* fd = fopen(path,"r");
        int read;
        char* us=(char*)malloc(45);
        char* pas=(char*)malloc(30);
        int len_us=0;
        int len_pas=0;
        if(fd<0)
        {
                printf("Eroare␣la␣deschidere␣fisier");
        }
        while((read=getline(&us,&len_us,fd))!=-1)
        {
```

```
                read= getline(&pas,&len_pas,fd);
                pas[strlen(pas)-1]='\0';
                us[strlen(us)-1]='\0';
                if(strcmp(password,pas)==0 && strcmp(username,us)==0)
                        {
                                return true;
                        }
        }
        return false;
}
```

5 Concluzii

Proiectul are mici lipsuri prin faptul ca o data ce se face un update de un intreg folder, fisierele de orice tip nu vor fi suprascrise, ci trebuie din nou executate comenzi de updateFile peste ele, pentru a contine ce contin de fapt pe masina clientului.
Un plus ar putea fi introducerea unei functii de autentificare pentru cei ce nu au deja un cont creat.
Totodata, separatorul pentru date e "∼", lucru care poate crea probleme cand se parseaza date(cum ar fi un fisier .txt) care contine un text in care am scris simbolul "∼". Trimiterea datelor pe rand (size, apoi buffer-ul de date) ar fi o alternativa mult mai buna.

6 Bibliografie

1. https://stackoverflow.com/questions/5467725/how-to-delete-a-directory-and-its-contents-in-posix-c/5495779
2. https://stackoverflow.com/questions/238603/how-can-i-get-a-files-size-in-c
3. hhttps://stackoverflow.com/questions/11790822/reading-a-large-file-using-c-greater-than-4gb-using-read-function-causing-pro
4. https://stackoverflow.com/questions/1786532/c-command-line-password-input/1786628
5. https://profs.info.uaic.ro/∼gcalancea/Laboratorul_7.pdf
6. https://profs.info.uaic.ro/∼gcalancea/laboratories.html
7. https://profs.info.uaic.ro/∼gcalancea/index.html
8. https://stackoverflow.com/questions/3501338/c-read-file-line-by-line