



Presentazione dell'esercitazione 4

Socket in C con select Server Multiservizio

Alice Turrini - Serena Bertaccini – Anna Vandi – Caterina Leonelli

Anno accademico 2021/2022

Obiettivi

Client datagram:

Inviare il nome file e la parola da eliminare all'interno dello **stesso datagramma**

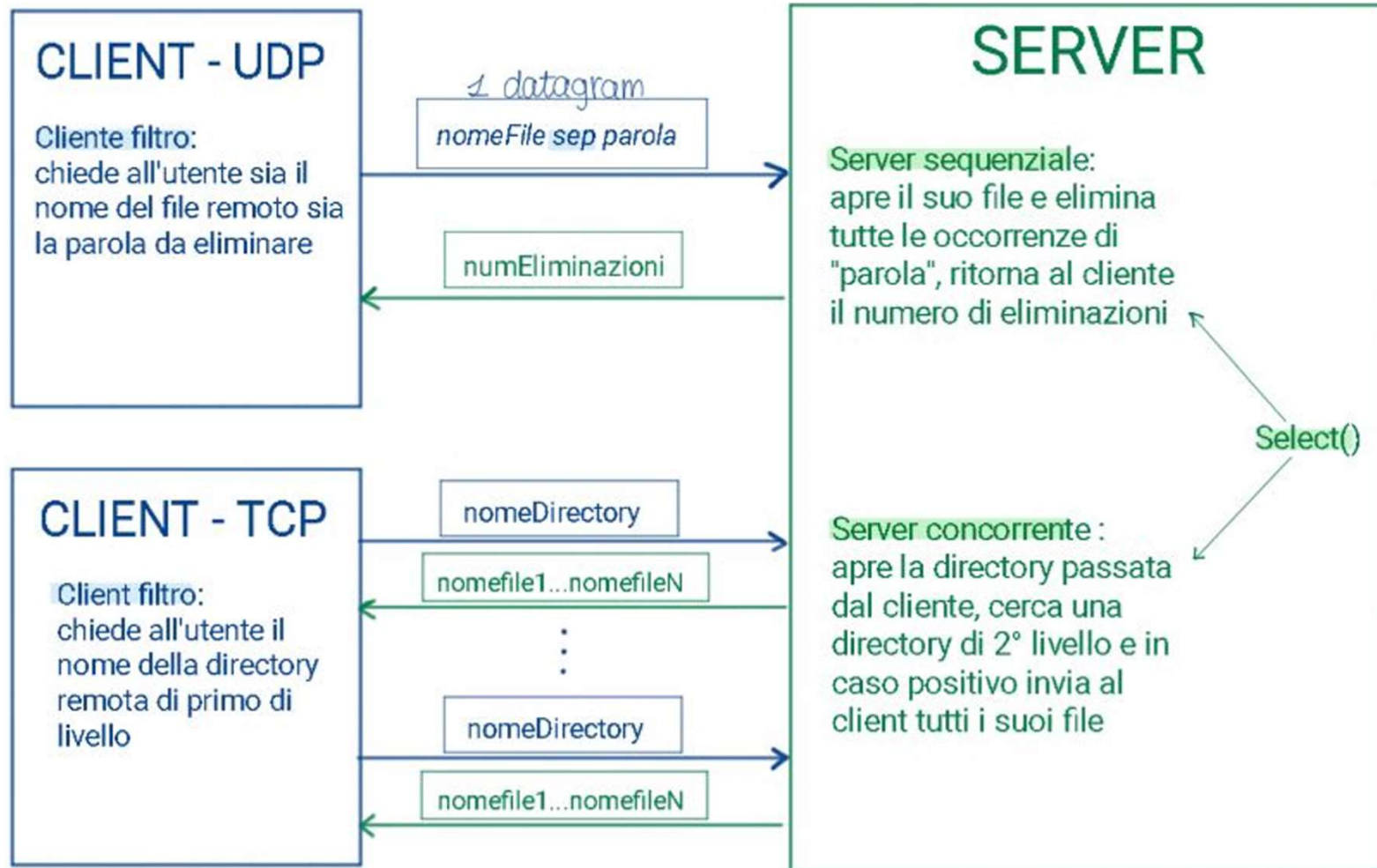
Client stream:

Usare **un'unica connessione per tutte le directory** richieste dall'utente

Server:

Il server deve **basarsi sulla *select*** e dovrà avere il protocollo allineato con quello del cliente sia datagram che stream


Struttura




Client

UDP (eliminazione parola)

```
while (gets(buff) != NULL) {
    strcat(buff, sep);
    printf("\nInserisci parola");
    gets(temp);
    strcat(buff, temp);

    if (sendto(sd, buff, ... )<0){
 printf("Errore! Nome nuovo file: ");
        continue;}

    if (recvfrom(sd, &num,...)<0){
 printf("Errore! Nome nuovo file: ");
        continue;}


    //controllo risultato
    if (num<0)
        printf("Errore nel file");
    else printf("Eliminati: %u\n", num);

    printf("Inserisci il nome del file: ");
} //fine while
```

sep: separa
i campi del
datagram

TCP (file dir II livello)

```
while (gets(dir) != NULL) {
    //invio nome directory al server
    if (write(sd, dir, (strlen(dir)+1))<0) {
        perror("Errore");

    }else{
 while(n=read(sd, buff, sizeof(buff)>0) {
        write(1, buff, strlen(buff));
    }
    if(n<0) perror("Lettura da stream");

    //chiedo ad utente un nuovo nome dir
    printf("Inserire nome directory: ");
}

//chiusura connessione
```

Utilizzo di una **stessa
connessione per
tutte le directory
richieste** dall'utente

Server select - UDP

for infinito,
demone

```
if (select(maxSd, &rset, NULL, NULL, NULL)<0){...}
if (FD_ISSET(udpSd, &rset)){ //gestione richiesta UDP
    if (recvfrom(udpSd, &line, ...) < 0){...}

    //sistema linea letta
    while(line[i]!=sep){nomeFile[i]=line[i]; i++;}
    nomeFile[i++]='\0';
    while(line[i]!='\0'){parola[j++]=line[i++];}
    parola[j]='\0';

    if ((fdFile=open(nomeFile, O_RDONLY))<0){...} //apro il file
    else{ temp_file="temp.txt"

        if((temp=open(temp_file, O_CREAT | O_RDWR, 0777))<0){...}

        else{ while(read(fdFile, &c, 1) > 0){

            if(c!=' ' && c!='\n'){word[i++]=c;} //creo le singole parole
            else{ word[i]='\0';

                if(strcmp(word, parola)!=0){ //comparo
                    write(temp, word, strlen(word)+1)<0) //scrivo solo se #
                }else numEliminated++;

                i=0; strcpy(word, "");
            }
        }

        sendto(udpSd, &numEliminated,...);
```



line: nomeFile+sep+parola





Server select - TCP

for infinito,
demone



```
if (select(maxSd, &rset, NULL, NULL, NULL)<0){...}
if (FD_ISSET(tcpSd, &rset)){//gestione richiesta TCP
    if((connfd = accept(tcpSd, ... ))<0){...}

    //figlio per la gestione della richiesta
    if (fork()==0){
        close(tcpSd); //chiudo socket lettura
        getcwd(dirServer, DIM_STRING);

        while((read(connfd, &nome_dir, sizeof(nome_dir)))>=0){ //lettura stream
            if((dir = opendir(nome_dir))==NULL){
                write(connfd,error,sizeof(error));}
            else {
                while ((dd = readdir(dir)) != NULL){ //scorro dir I livello
                    if(dd->d_type == DT_DIR && strcmp(dd->d_name, ".")!=0 &&
                       strcmp(dd->d_name, "..")!=0){
                        getcwd(path1, DIM_STRING); strcat(path1, "/");
                        strcat(path1, nome_dir); chdir(path1); //mi sposto
                        while ((dd2 = readdir(dir2)) != NULL){ //scorro dir II livello

                            if(dd2->d_type == DT_REG){
                                write(connfd, dd2-> d_name , sizeof(dd2-> d_name)); //invio
                                write(connfd, dd2-> d_name , sizeof(dd2-> d_name));
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Esecuzione



Client Datagram

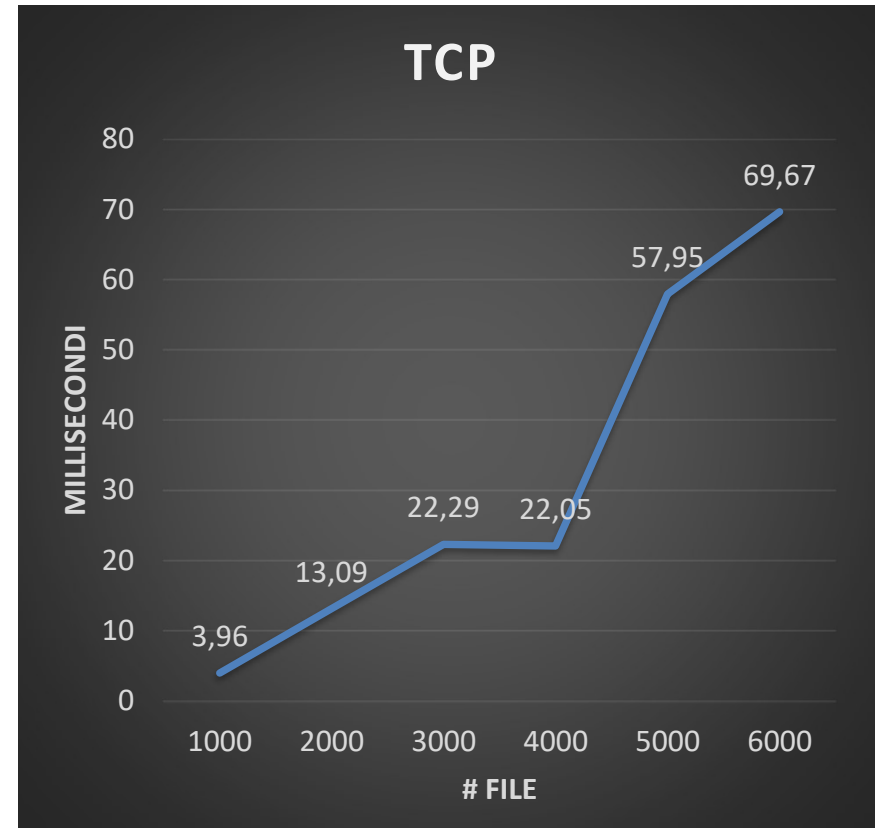
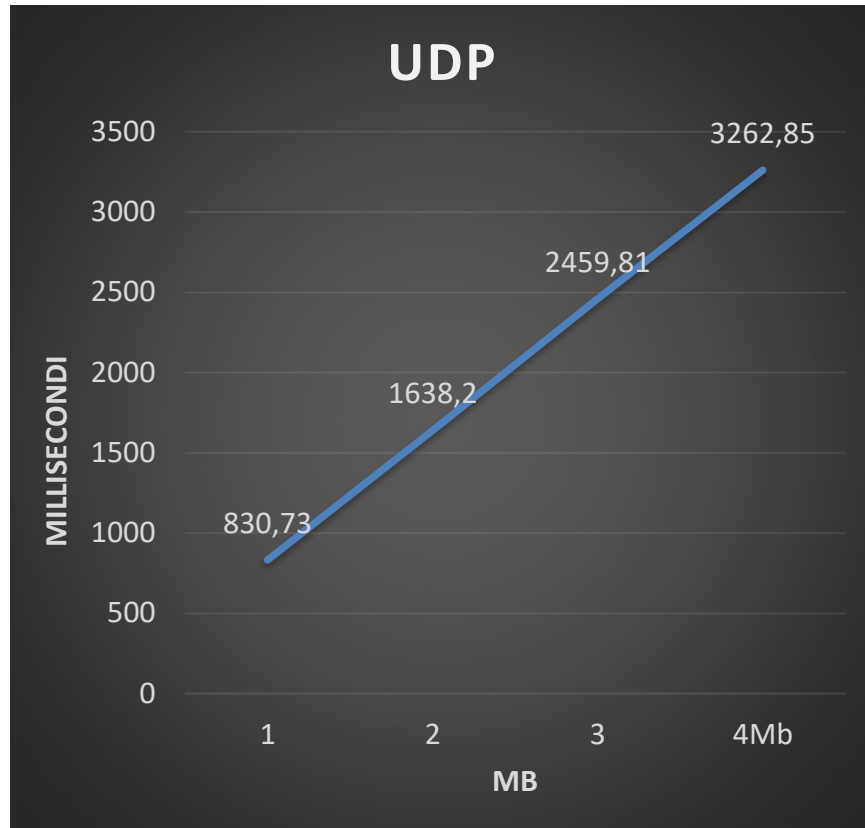
```
--Client avviato--  
Creata la socket sd=3  
Client: fatta bind socket alla porta 0  
  
Inserisci nome del file: prova.txt  
|prova.txt?|  
Inserisci la parola da eliminare sul file: fare  
Invio datagram: prova.txt?fare  
  
...Attesa del risultato...  
Numero eliminazioni nel file: 0  
Inserisci il nome del file: prova.txt  
|prova.txt?|  
Inserisci la parola da eliminare sul file: fare  
Invio datagram: prova.txt?fare  
  
...Attesa del risultato...
```

```
Numero eliminazioni nel file: 3  
Inserisci il nome del file:  
Client: termino...
```

Server Datagram

```
Ho letto il nome del file: prova.txt  
Ho letto parola: fare  
HO APERTO IL FILE  
HO APERTO TEMP  
  
da scrivere tutti  
da scrivere quanti  
da scrivere voglion  
da scrivere Jazz  
da scrivere tutti  
da scrivere quanti  
da scrivere voglion  
da scrivere Jazz  
da scrivere tutti  
da scrivere quanti  
da scrivere voglion
```

Tempistiche



Conclusioni

Per mandare **un solo datagram con due informazioni** si è fatto uso di un carattere separatore

La **select** è stata fondamentale per distinguere i due tipi richieste provenienti dai due tipi di client

Necessario l'uso **ottimizzato** e ridotto delle risorse, soprattutto quando si lavora nel distribuito