Gruppo 17

Università degli Studi di Bologna
Scuola di Ingegneria

Esercitazione 3

Conteggio Parole/Elimina Linea
Socket C senza e con connessione

Corso di Reti di Calcolatori T

Daniele Magagnoli, Giulia Martina Bal,
Antonio Cassanelli, Gianmiriano Porrazzo

Anno Accademico 2021/2022

SPECIFICHE DELL'ESERCITAZIONE (PRIMA PARTE)

- L'esercitazione si divide in due parti : la prima per affrontare le socket senza connessione, mentre nella seconda useremo le socket con connessione.
- Nella parte di uso delle socket senza connessione, dobbiamo sviluppare un'applicazione Client/Server che restituisce la lunghezza della parola più lunga presente in un file situato sul filesystem di un Server remoto.

MAX WORD CLIENT

```
printf("Inserire il nome di un file, EOF per terminare: ");

while ( gets(nomeFile) != EOF ) //oppure senza EOF??
{
    printf("Nome file letto: %s\n", nomeFile);

    /* richiesta operazione */
    len=sizeof(servaddr);
    if(sendto(sd, nomeFile, sizeof(nomeFile), 0, (struct sockaddr *)&servaddr, len)<0){
        perror("sendto");
        continue;
    }

    /* ricezione del risultato */
    printf("Attesa del risultato...\n");
    if (recvfrom(sd, &ris, sizeof(ris), 0, (struct sockaddr *)&servaddr, &len)<0){
        perror("recvfrom"); continue;}

    if (ris<0)
        printf("File non presente nel server \n");
    else
        printf("Esito dell'operazione: %d\n", ris);

    printf("Inserire il nome di un file, EOF per terminare: ");
} // while gets
```

Il client si comporta come un filtro, chiedendo all'utente ad ogni iterazione il nome di un file presente sul file system del server a cui invia il nome del file cercato.

Per fare ciò leggiamo da input il nome del file finché questo non si inserisce ^D(UNIX)/^Z(Windows) ed usiamo una sendto per inviare il nome del file, visto che usiamo una connessione datagram.

Riceviamo infine la risposta usando la recvfrom nella quale viene specificato il numero di lettere che compongono la parola più lunga all'interno del file specificato.

MAX WORD SERVER

```
for(;;){
    max=0;
    len=sizeof(struct sockaddr_in);
    if (recvfrom(sd,nomeFile, sizeof(nomeFile), 0, (struct sockaddr *)&cliaddr, &len)<0)
    {perror("recvfrom "); continue;}



    if ((fd = open(nomeFile, O_RDONLY)) < 0) {
        perror("File inesistente");
        max=-1;
    }else{

        start=clock();
        while ((n=read(fd, &ch ,sizeof(char)))>0) {

            if(ch==' ' || ch=='\n') {
                if(count>max)
                    max=count;
                count=0;
            }else
                count++;
        }
        end=clock();
    }
    close(fd);
    clienthost=gethostbyaddr( (char *) &cliaddr.sin_addr, sizeof(cliaddr.sin_addr), AF_INET);
    if (clienthost == NULL) printf("client host information not found\n");
    else printf("Operazione richiesta da: %s %i\n", clienthost->h_name,(unsigned)ntohs(cliaddr.sin_port));

    printf("Tempo di escuzione byte per byte: %f\n", (float)(end-start)/CLOCKS_PER_SEC);

    if (sendto(sd, &max, sizeof(max), 0, (struct sockaddr *)&cliaddr, len)<0)
    {perror("sendto "); continue;}
} //for
```

Il server attende fin quando non riceve nome di un file da un cliente.
Una volta ricevuto il nome del file verifica che sia presente sul proprio file system. Nel caso in cui il file non esista server restituisce -1, altrimenti cerca la parola più lunga e la restituisce al cliente.

L'algoritmo di ricerca della parola più lunga prevede di leggere ogni carattere del file e fermarsi non appena si trova '\n' o uno spazio salvando la lunghezza della parola letta.

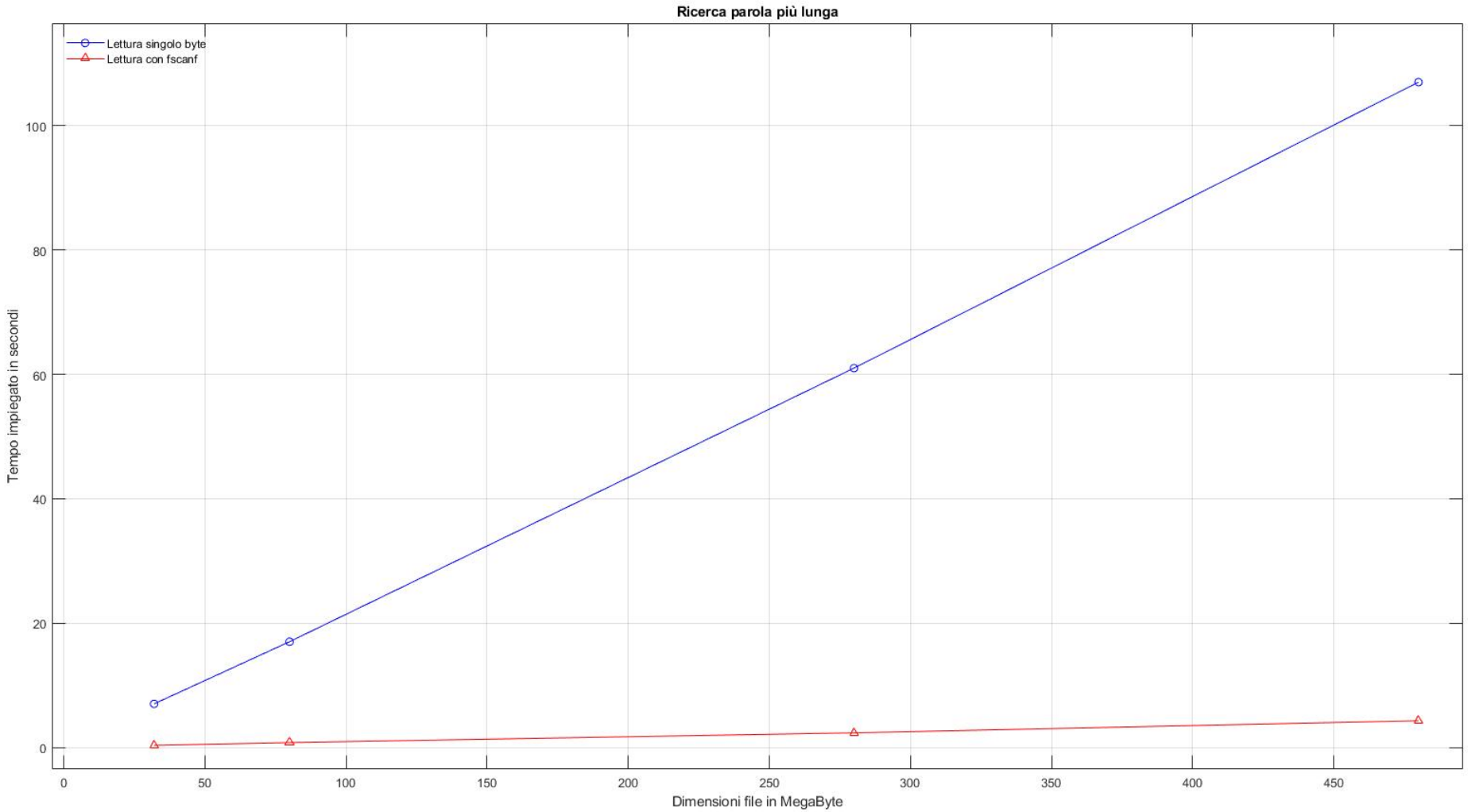
Infine, se tale lunghezza supera quella salvata nella variabile max, questa si aggiorna altrimenti si continua a cercare una parola più lunga nel file.

MAX WORD SERVER (SOLUZIONE ALTERNATIVA)

```
while (fscanf(fd, "%s", &parola)>0) {

if (strlen(parola) > max)
    max = strlen(parola);
}
```

Leggere un file a carattere non è la cosa migliore nella vita, per questo abbiamo pensato ad una soluzione alternativa basata sul leggere il file parola per parola in modo da poter contare la dimensione della parola e confrontarla con quella presente nella variabile max.

Per fare ciò abbiamo usato la funzione fscanf, la quale legge fino al prossimo spazio o ‘\n’.


Ricerca parola più lunga
Lettura singolo byte
Lettura con fscanf
Tempo impiegato in secondi
0
20
40
60
80
100
0
50
100
150
200
250
300
350
400
450
Dimensioni file in MegaByte

Specifiche dell’esercitazione (seconda parte):

- Nella seconda parte dell’esercitazione ci viene chiesto di sviluppare un’applicazione Client/Server per eliminare una riga, specificata dall’utente, di un file presente sul filesystem del Client.
- Il Client dunque dovrà chiedere all’utente la riga da eliminare e il nome del file da cui cancellarla.
- Il server (concorrente) si occuperà di eliminare la riga specificata e inviare il file finale al client, senza salvare una copia in locale.

DELETE SERVER

```c
if (fork()==0){ // figlio
    /*Chiusura FileDescr non utilizzati e ridirezione STDIN/STDOUT*/
    close(listen_sd);
    host=gethostbyaddr( (char *) &cliaddr.sin_addr, sizeof(cliaddr.sin_addr), AF_INET);
    if (host == NULL){
        printf("client host information not found\n");
        close(conn_sd);
        exit(2);
    }
    else printf("Server (figlio): host client e' %s \n", host->h_name);

    //Leggo il numero della linea
    if((read(conn_sd,&numLine,sizeof(int))>0))
        printf("Server (figlio), linea da eliminare: %d\n",numLine);

    //inizializzo il numero della linea a 1 per indicare che
    //leggo la prima linea
    lcount=1;

    //Sostituzione delle righe
    while((nread=read(conn_sd,&car,sizeof(char)))!=0){
        if(nread<0){
            sprintf(error,"(PID %d) impossibile leggere dal file",getpid());
            perror(error);
            exit(EXIT_FAILURE);
        }
        //leggo e scrivo 1 byte alla volta, posso fare meglio?
        if(car=='\n'){
            //printf("Linea %d terminata \n",lcount);
            lcount++;
            if(lcount!=numLine)
                write(conn_sd,&car,1);
        }else{
            if(lcount==numLine) //non trasmetto la linea
                //printf("Linea da eliminare\n");
            else
                write(conn_sd,&car,1);
        }
    }
```

Per usare un server concorrente facciamo in modo che il processo padre crei un processo figlio per ogni richiesta.

Il figlio si occuperà di trovare la linea da eliminare leggendo il file byte per byte dalla connessione in modo da distinguere una linea dall'altra quando troviamo il carattere '\n'.

Il file sarà ritrasmesso al cliente senza la linea da eliminare.

DELETE CLIENT

Il client, una volta inviato il file al server, rimane in attesa che il server mandi il file modificato, in modo da poterlo salvare sul proprio file system e stamparlo a video.

Per fare ciò abbiamo pensato in prima battuta di fare degli invii / delle ricezioni byte per byte, ciò risulta essere poco efficiente.

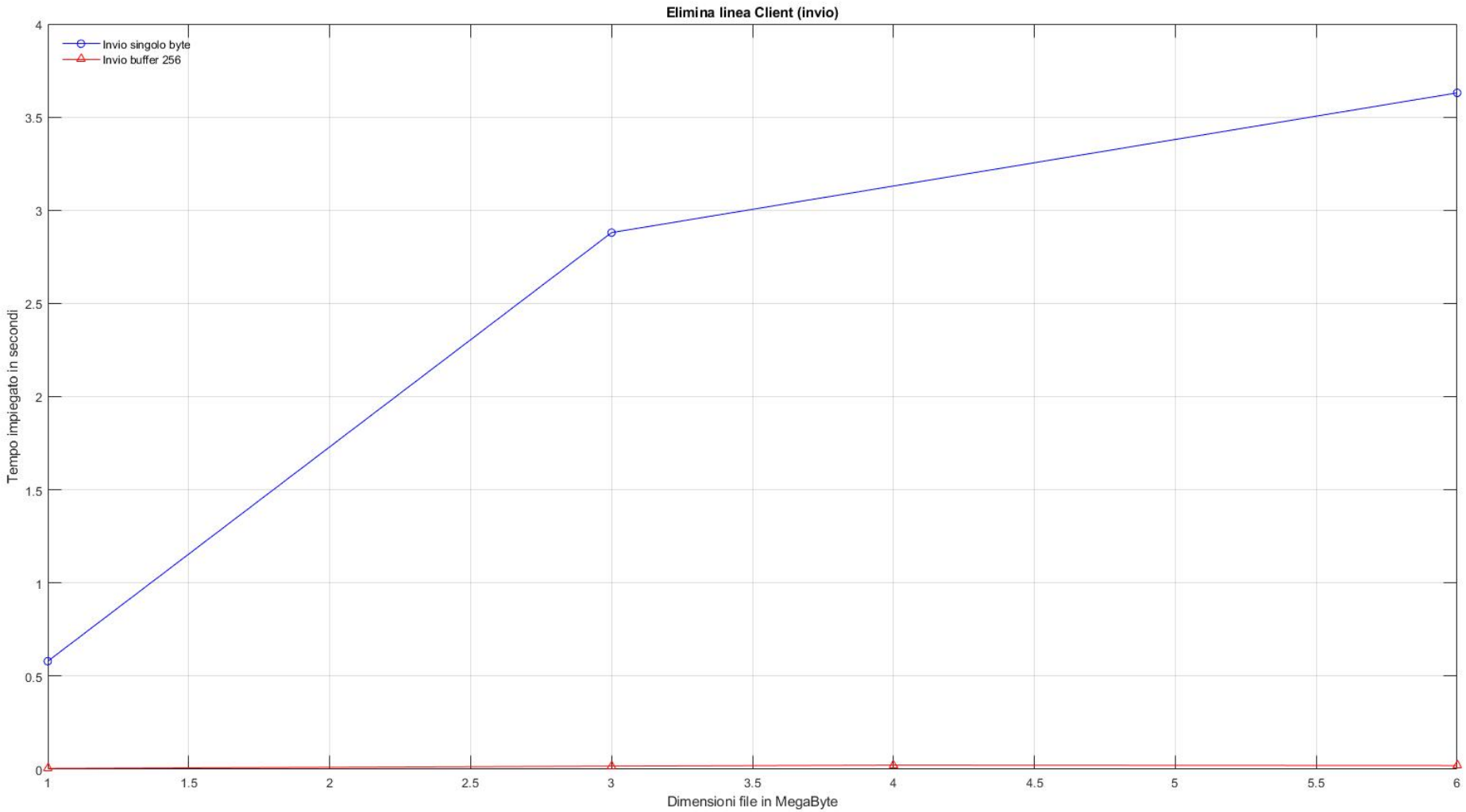
```
/*INVIO File byte per byte*/
printf("Client: invio file\n");

while(nread = read(fd_file, &buff, 1) != 0){
    write(sd, &buff, 1);
}
```

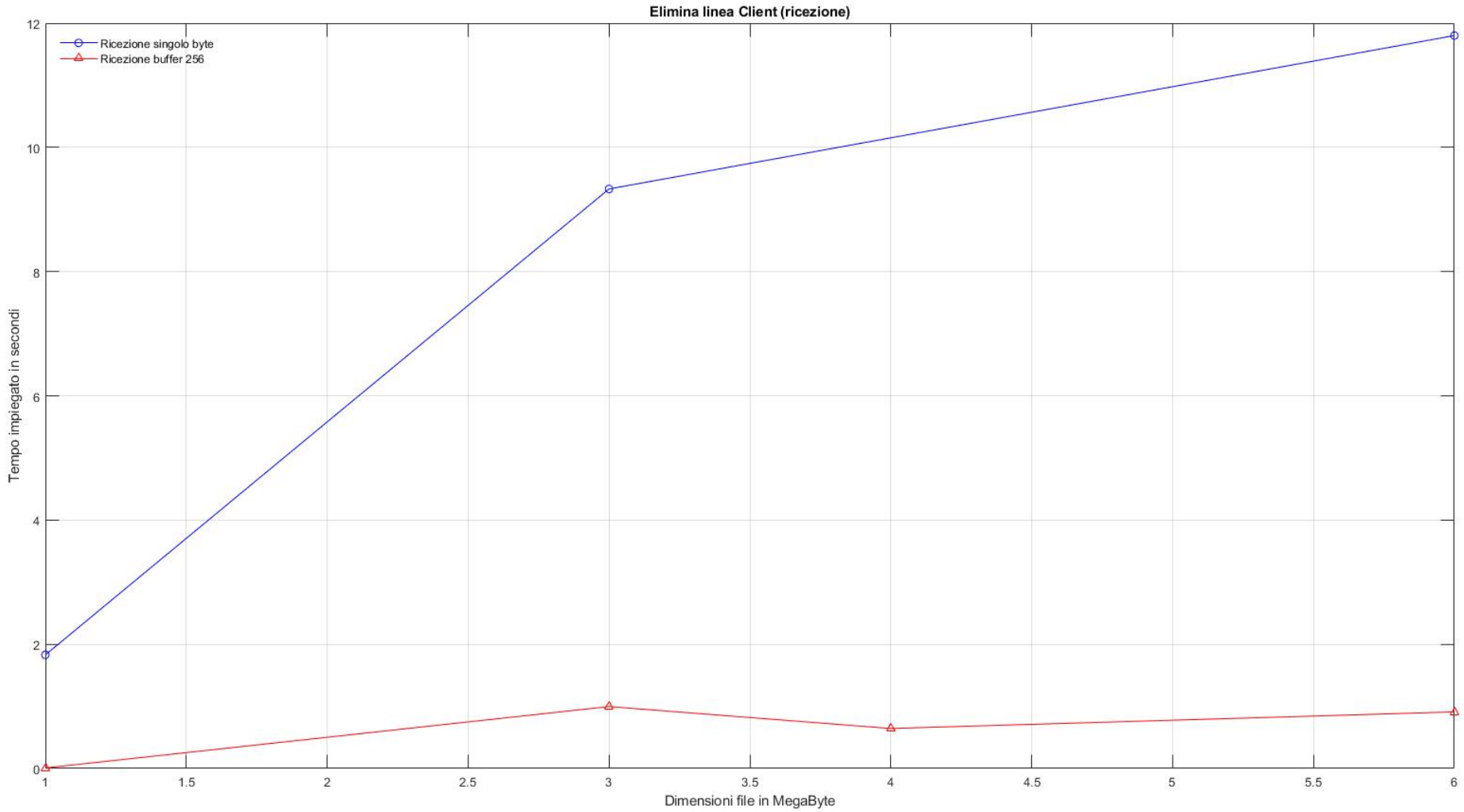
```
/*RICEZIONE File  byte per byte*/
printf("Client: ricevo e stampo file ordinato\n");

while((nread=read(sd,&buff,1))>0)
{
    write(fd_file, &buff, 1);
    write(1, &buff, 1);
}
```

```
/*INVIO File con buffer*/
printf("Client: invio file \n");

while((nread = read(fd_file, buff, DIM_BUFF)) > 0){
    write(sd, buff, nread);
}
```

```
/*RICEZIONE File con buffer*/
printf("Client: ricevo e stampo il file senza linea\n");

while((nread=read(sd,buff,DIM_BUFF))>0)
{
    write(fd_dest, buff, nread);
    write(1, buff, nread);
}
```

Anche in questo caso abbiamo pensato ad una soluzione alternativa che prevede l'uso di un buffer di dimensione 256 byte, sia per l'invio del file che per la ricezione del file modificato.


Elimina linea Client (invio)
Invio singolo byte
Invio buffer 256
Tempo impiegato in secondi
Dimensioni file in MegaByte
0
0.5
1
1.5
2
2.5
3
3.5
4
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6


Elimina linea Client (ricezione)
Ricezione singolo byte
Ricezione buffer 256
Tempo impiegato in secondi
Dimensioni file in MegaByte
0
2
4
6
8
10
12
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

GRAZIE PER L'ATTENZIONE