ESERCITAZIONE 3

Socket in C senza e con connessione

- Bernardi Daniel
- Chichifoi Karina
- Gjura Endri
- Ivan Andrei Daniel
- Pizzini Cavagna Hiari

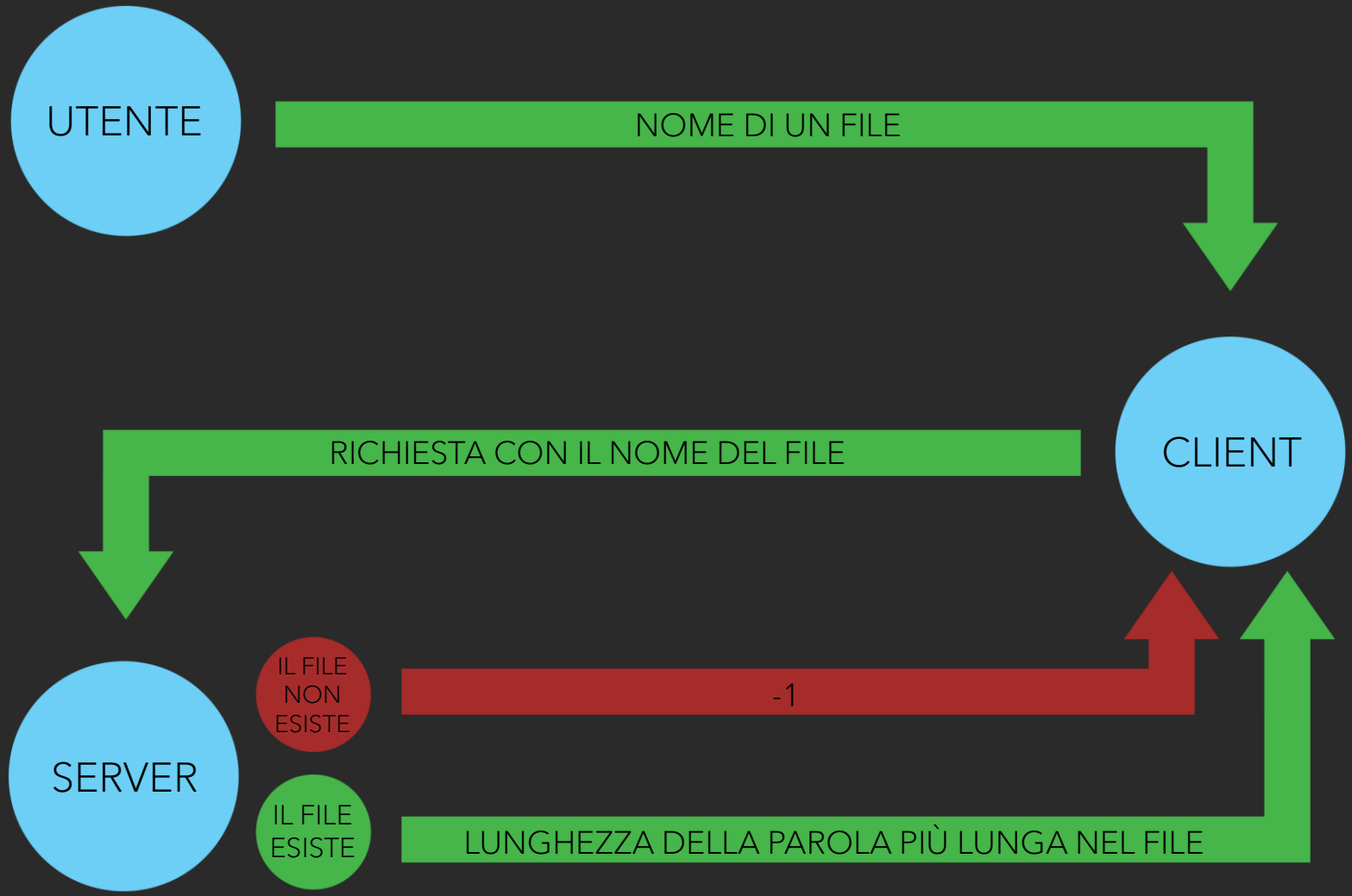
Introduzione

Il Server UDP è stato sviluppato inizialmente in versione sequenziale, poi è stata creata anche la versione parallela puntando a delle performance migliori.
Utilizzando un cronometro nel codice si è cercato di misurare il tempo di risposta del Server per confrontare entrambe le versioni; tuttavia sono incorsi problemi nell'ottenere dei risultati attendibili che mettessero in evidenza l'efficienza dell'uno rispetto all'altro.

Per quanto riguarda il riconoscimento delle parole il Server considera come separatore qualsiasi carattere diverso dalle lettere dell'alfabeto. Sarebbe stato possibile implementare l'aggiunta di un ulteriore argomento per definire una lista di separatori personalizzata.

Il Server TCP è stato prodotto con due algoritmi di lettura differenti per testare qual è il più performante: uno legge carattere per carattere dalla socket, l'altro legge con un buffer di 256 caratteri.

Schema C/S senza connessione (UDP)



Codice Client UDP (1)

```
printf("Inserire il nome di un file o EOF (CTRL + D) per terminare: ");

while (gets(fileName)) {
        nameLength = strlen(fileName);
        if (nameLength > 4
            && fileName[nameLength-4] == '.'
            && fileName[nameLength-3] == 't'
            && fileName[nameLength-2] == 'x'
            && fileName[nameLength-1] == 't') {

            // Copio l'array
            memcpy(&(request.file), &fileName, sizeof(request.file));

            length = sizeof(servaddr);
            if (sendto(socketDescriptor, &request, sizeof(Request), 0, (struct sockaddr*)&servaddr, length) < 0) {
                perror("Errore nella sendto.");
                continue;
            }
```

Si prende dall'input il nome del file che l'utente passa e si controlla che sia un file di testo (formato .txt); se non lo è, non si fa nulla e si stampa una riga apposita (si veda slide successiva)

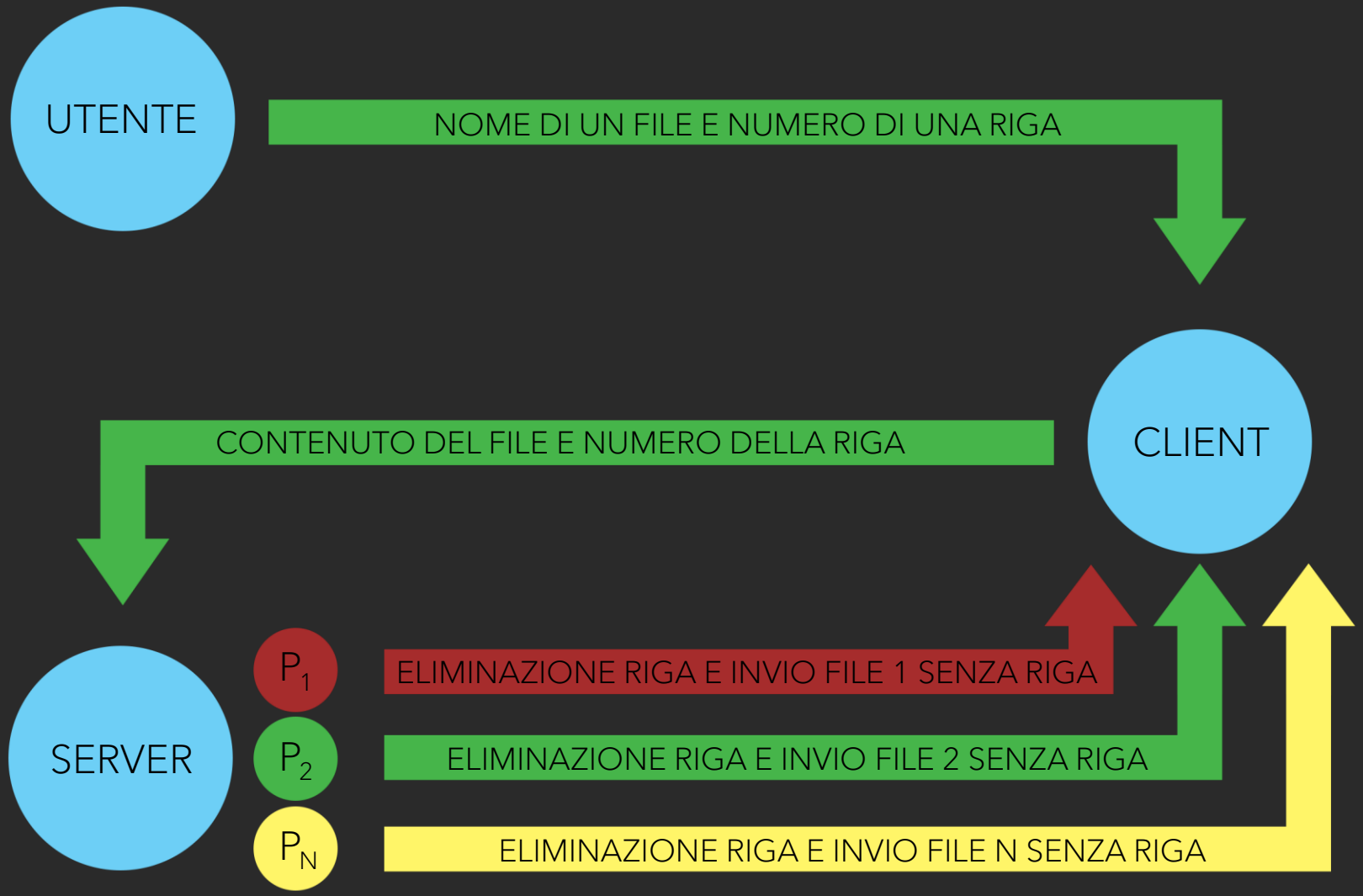
Codice Client UDP (2)

```
        if (recvfrom(socketDescriptor, &result, sizeof(result), 0, (struct sockaddr*)&servaddr, &length) < 0) {
            perror("Errore nella recvfrom.");
            continue;
        }

        if ((int)ntohl(result) > 0) {
            printf("La parola piu' lunga nel file richiesto ha %i caratteri.\n", (int)ntohl(result));
        } else if ((int)ntohl(result) == 0) {
            printf("Nel file richiesto non ci sono parole.\n");
        } else {
            printf("Il file %s non esiste sul server.\n", fileName);
        }
    } else {
        printf("Il file inserito non è un file di testo (*.txt).\n");
    }

    printf("Inserire il nome di un file o EOF (CTRL + D) per terminare: ");
}
```

Si gestiscono i tre casi di ciò che si può ricevere dal server:

- viene trovata la parola più lunga e si stampa il numero di caratteri;
- se si riceve 0, si stampa che nel file richiesto (che esiste) non ci sono parole;
- se invece si riceve -1 significa che il file richiesto non esiste sul server.

Schema C/S con connessione (TCP)



Codice Server TCP (1)

```
while ((readSocket = read(newSocket, &c, sizeof(char))) > 0) {
    if (numLinea != contaLinea) {
        write(newSocket, &c, sizeof(char));
    }
    if (c == '\n') {
        contaLinea++;
    }
}
```

Variante dell'algoritmo per leggere e scrivere sulla socket usando un carattere alla volta.

Codice Server TCP (2)

```
while ((readSocket = read(newSocket, &buff, DIM_BUFF)) > 0) {
     i = 0;
     while (i < readSocket) {
          if (numLinea != contaLinea) {
               write(newSocket, &(buff[i]), sizeof(char));
          }
          else {
               while(buff[i] != '\n' && i < readSocket) {
                    i++;
               }
          }

          if (buff[i] == '\n') {
               contaLinea++;
          }

          i++;
     }
}
```

Variante dell'algoritmo per leggere e scrivere sulla socket usando un buffer da 256 caratteri.

Conclusione

I test di performance dei due algoritmi usati nel Server TCP hanno evidenziato una diminuzione del tempo di risposta del Server, per il singolo Client, nel caso di lettura dalla socket con buffer da 256 caratteri.

	Buffer da 1 carattere	Buffer da 256 caratteri
Lettura e scrittura su Socket in TCP	~105 ms	~60 ms