Relazione progetto Nim

Per il seguente progetto, realizzato mediante il linguaggio C, si doveva implementare una versione del gioco Nim che si basasse su una connessione client/server locale.

Il nostro gruppo ha implementato il gioco nel seguente modo:

- nimserver, gestisce la connessione dei client, una volta connessi entrambi i player, la partita verrà generata tramite la creazione del campo da gioco costituito dalle 2 pile e un numero di casuale di pedine per ciascuna.
- nimclient permette al client di collegarsi alla lista d'attesa per una partita di nimserver.
- client si occupa gestisce le funzioni del client, ovvero la connessione al server, la ricezione di messaggi da parte del server, del controllo del vincitore ad ogni turno e invio degli input.
- server si occupa delle funzioni del server, ovvero l'attesa della connessione dei client, la creazione del campo da gioco, l'inizio della partita, l'invio di messaggi al client, la gestione del turno, la gestione degli input del client e della terminazione della partita in caso della ricezione di un vincitore.
- tools è una raccolta di varie funzioni per la gestione dei socket e dei risultati sia per il client che per il server.
- game per le funzioni riguardanti la partita, cioè la creazione di un campo da gioco, la scelta di una pila e del numero di pedine da rimuovere, il cambio turno la decretazione del vincitore.

Il progetto è stato testato con successo sulle seguenti macchine:

1)

Sistema operativo: MANJARO Linux KDE 18.1.5

Versione kernel: Linux 5.4.13 x86_64

Versione compilatore: clang 9.0.1

2)

(Virtual Machine)

Sistema operativo: Elementary OS 5.1 Hera (basato su Ubuntu)

Versione kernel: Linux 4.15.0 x86_64

Versione compilatore: clang 6.0.0

GESTIONE ERRORI

Da specifiche era stato richiesto che il programma fosse il più robusto possibile, in particolare in caso di disconnessione di uno o più client dal server, questo non doveva interrompersi ma semplicemente chiudere la partita in corso.

La prima "misura di sicurezza" adottata è stata quella di utilizzare *signal(SIGPIPE, SIG_IGN)* sia su nimserver che su nimclient, in questo modo, in caso di chiusura "forzata" di una connessione (ambo i lati), i programmi semplicemente ignorano l'errore (che viene comunque riconosciuto e gestito adeguatamente). In secondo luogo, all'inizio di nimserver viene chiamata la funzione *unlink* sull'indirizzo del socket file utilizzato (che per comodità è indicato come una costante).

Dai primi test sono risultati tre casi di errore differenti da gestire:

1. Il server chiude la connessione: in questo caso entrambi i client riconoscono l'errore, chiudono la connessione, segnalano l'errore (stampandolo su stderr) e chiudono il programma;
2. Il client non di turno chiude la connessione: il client di turno conclude la sua mossa, ma a fine turno viene lanciato il signal SIGPIPE (perché il server chiude entrambe le connessioni coi giocatori), segnalato l'errore, chiusa la connessione al server e chiuso il programma;
3. Il client di turno chiude la connessione: in questo caso il server chiude immediatamente le connessioni concludendo la partita, tuttavia dai test effettuati è risultato che dal lato del client rimasto collegato non viene segnalato alcun errore, ma la funzione recv ritorna sempre 0 come valore, per cui adesso viene segnalato l'errore anche quando vengono ricevuti 0 byte dal server/client (dato che utilizzano lo stesso protocollo). La gestione di questo errore è identica a quella del punto 2.

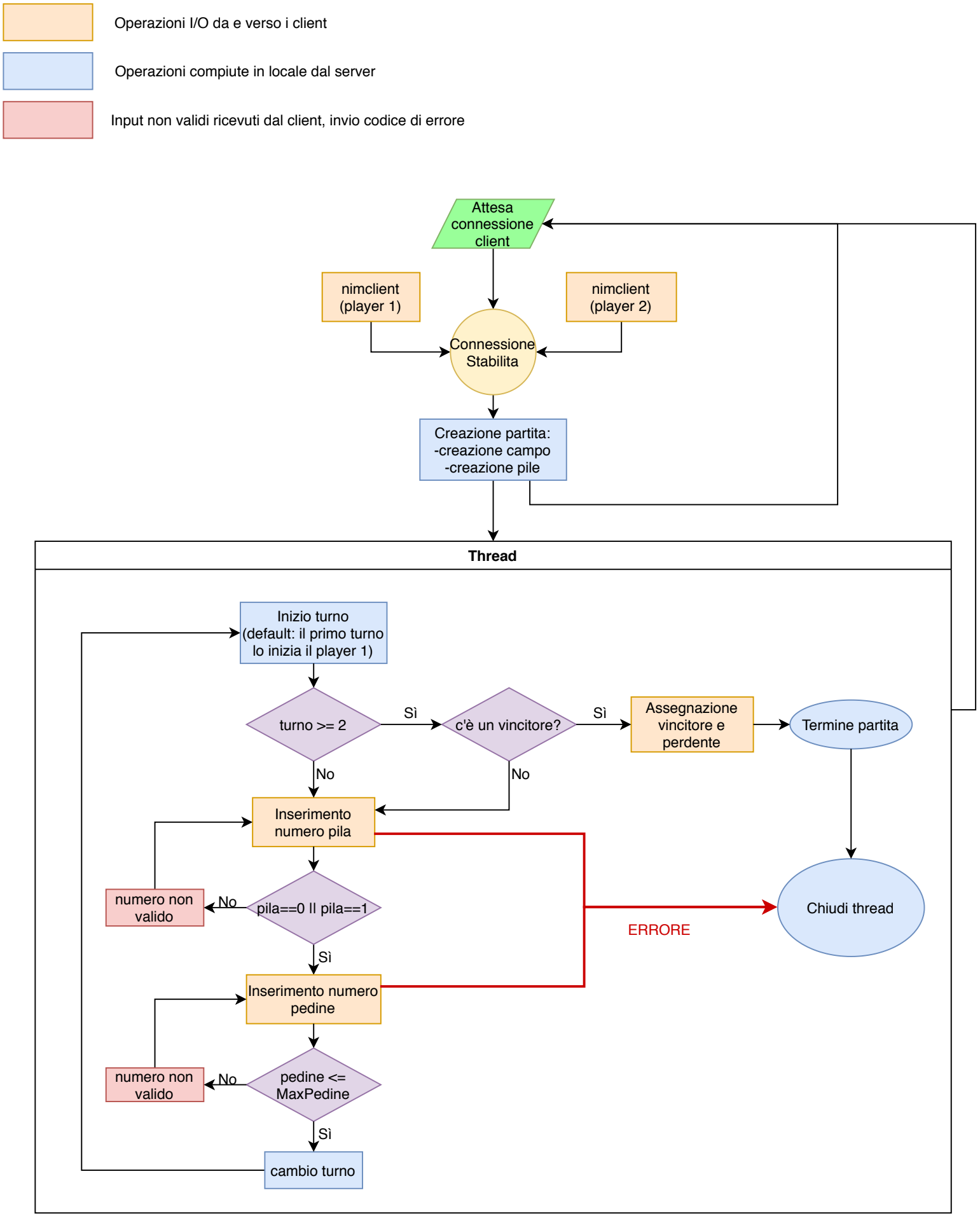
All'interno del nostro programma, per coerenza e una più corretta gestione del programma, abbiamo distinto e contrassegnato gli errori più comuni:

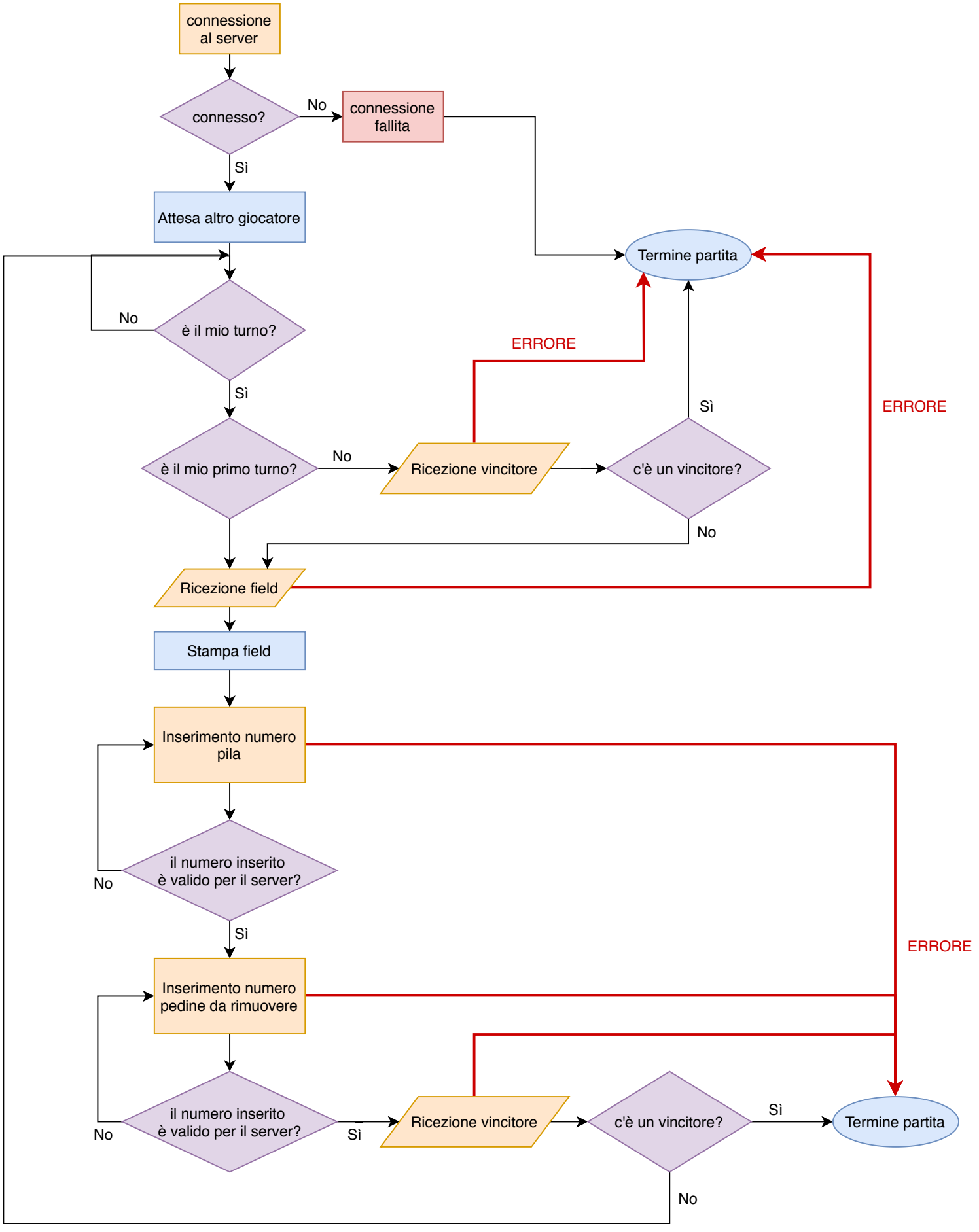
1. Socket invalido: chiamato durante la creazione di un socket (se questo non viene creato);
2. Errore: quando viene rifiutata una connessione
3. Disconnessione: per qualsiasi altro errore tra client e server

Nel caso di errato inserimento dei valori da parte del player, sia per la pila da scegliere che per il numero di pedine da voler rimuovere, viene ritornato INVALID_RANGE e il client stampa su stdout un avviso di errata immissione dei parametri, permettendo successivamente al player il reinserimento dei valori, fino alla trasmissione di parametri idonei.

Durante la gestione del turno nel client, sia a inizio che fine, abbiamo provveduto a controllare la presenza del vincitore restituendo NO_WINNER se non presente, altrimenti ritornando il codice del player che ha vinto.

Il funzionamento è riassunto nei seguenti diagrammi di flusso:


LEGENDA
Operazioni I/O da e verso i client
Operazioni compiute in locale dal server
Input non validi ricevuti dal client, invio codice di errore
Attesa connessione client
nimclient (player 1)
nimclient (player 2)
Connessione Stabilita
Creazione partita: -creazione campo -creazione pile
Thread
Inizio turno (default: il primo turno lo inizia il player 1)
turno >= 2
Sì
c'è un vincitore?
Sì
Assegnazione vincitore e perdente
Termine partita
No
No
Inserimento numero pila
numero non valido
No
pila==0 || pila==1
Sì
ERRORE
Chiudi thread
Inserimento numero pedine
numero non valido
No
pedine <= MaxPedine
Sì
cambio turno


connessione al server
connesso?
No
connessione fallita
Sì
Attesa altro giocatore
Termine partita
No
è il mio turno?
ERRORE
Sì
Sì
ERRORE
è il mio primo turno?
No
Ricezione vincitore
c'è un vincitore?
No
Ricezione field
Stampa field
Inserimento numero pila
il numero inserito è valido per il server?
No
Sì
ERRORE
Inserimento numero pedine da rimuovere
il numero inserito è valido per il server?
No
Sì
Ricezione vincitore
c'è un vincitore?
Sì
Termine partita
No