Raport Tehnic: Arhitectura și Implementarea Sistemului Distribuit "QuizzGame"

Nica Rareș-Constantin

1. Introducere

Proiectul vizează dezvoltarea aplicației distribuite "QuizzGame", un sistem interactiv de tip Client-Server care gestionează sesiuni concurente de joc. Obiectivul central este coordonarea simultană a unui număr nedefinit de utilizatori, serverul având rolul de a sincroniza rundele de întrebări și de a gestiona starea globală a competiției.

Arhitectura centralizată conferă serverului autoritate totală asupra logicii jocului, acesta impunând limite stricte de timp pentru răspunsuri (sincronizare temporală) și gestionând dinamic intrările și ieșirile participanților. Sistemul este proiectat cu accent pe robustețe, izolând erorile cauzate de deconectările abrupte pentru a nu afecta experiența utilizatorilor rămași activi.

2. Tehnologii Aplicate

Pentru a satisface cerințele de performanță și concurență, soluția utilizează tehnologii native din ecosistemul Linux/UNIX:

• Limbaj și Rețea (C & TCP): Aplicația este implementată în C folosind API-ul POSIX pentru un control granular asupra resurselor. Comunicarea se realizează prin protocolul TCP, esențial pentru a garanta integritatea și ordinea corectă a pachetelor de date (întrebări, scoruri), eliminând riscurile asociate protocolului UDP în contexte critice.

• Concurență (POSIX Threads): Serverul adoptă un model multithreaded (<pthread.h>) în detrimentul proceselor (fork). Această alegere permite partajarea eficientă a memoriei între firele de execuție, facilitând accesul concurent și sincronizat (prin Mutex-uri) la resursele globale (lista de clienți, starea jocului).

• Persistență (SQLite): Gestionarea seturilor de întrebări este realizată prin motorul de baze de date SQLite. Acesta oferă o soluție robustă și structurată pentru stocarea datelor, decuplând conținutul static de logica de execuție a serverului și permițând interogări SQL eficiente la inițializare.

3. Structura Aplicației

Arhitectura sistemului este una centralizată, de tip Client-Server, unde serverul deține controlul absolut asupra logicii de desfășurare a jocului, iar clienții acționează ca terminale de intrare/ieșire.

3.1. Componenta Server

Serverul este o aplicație multithreading complexă, organizată pe două paliere de execuție distincte, care comunică prin memorie partajată protejată de un mecanism de sincronizare (Mutex):

1. **Nivelul de Conexiune și Interacțiune (Client Handlers):** Funcția main acceptă conexiuni într-o buclă infinită. Pentru fiecare client conectat, serverul lansează un fir de execuție dedicat (thread). Acest fir are rolul de a gestiona comunicarea directă cu clientul specific (primirea comenzilor LOGIN/QUIT, preluarea răspunsurilor la întrebări) și de a actualiza structura de date globală a jucătorului în memoria partajată.
2. **Nivelul de Coordonare Centrală (Game Manager):** Spre deosebire de arhitecturile clasice reactive, serverul rulează un fir de execuție independent (game_logic) care acționează ca un "dirijor". Acesta implementează mașina de stări a jocului:
    - **Lobby:** Monitorizează numărul de jucători activi și așteaptă atingerea pragului minim.
    - **Joc:** Extrage întrebările din baza de date, le trimite sincron tuturor clienților și gestionează cronometrul global (folosind un mecanism de polling secundă cu secundă pentru a detecta deconectările în timp real).
    - **Votare și Restart:** La finalul rundelor, coordonează procesul de votare (Y/N) pentru reînceperea jocului, eliminând clienții inactivi și resetând starea pentru o nouă sesiune.
3. **Nivelul de Date (Data Layer):** Interfațează cu baza de date SQLite (quiz.db) pentru a încărca setul de întrebări la pornirea aplicației, asigurând eficiența prin menținerea acestora în memoria RAM (set_intrebari) pe durata execuției.

3.2. Componenta Client

Clientul este o aplicație consolă care implementează Multiplexarea I/O folosind apelul de sistem select(). Această arhitectură permite clientului să monitorizeze simultan două surse de evenimente:

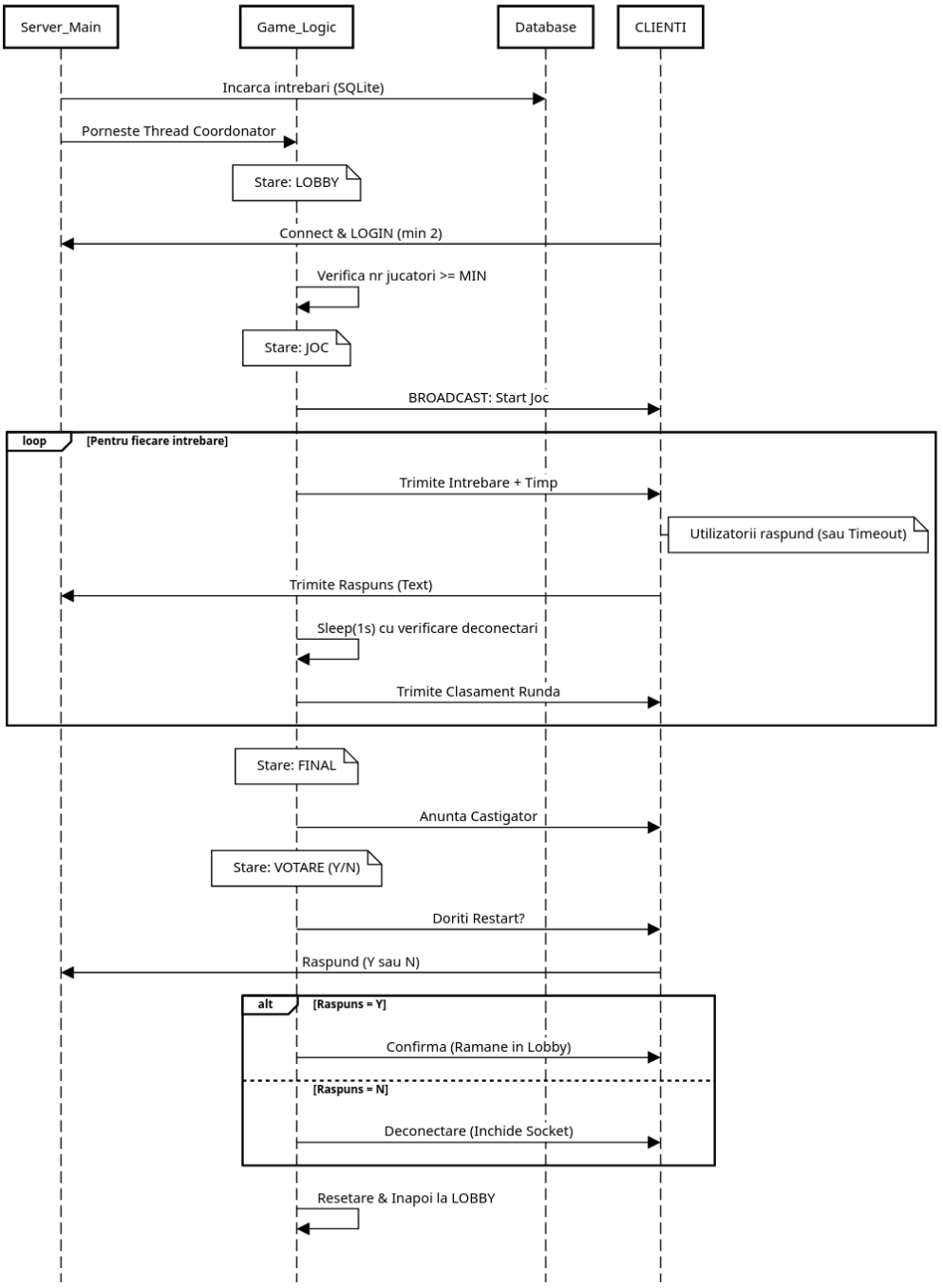
1. **Intrarea Standard (Tastatură)**: Pentru a prelua comenzile utilizatorului.
2. **Socket-ul de rețea**: Pentru a primi mesaje asincrone de la server (ex: întrebări, notificări de expirare a timpului, deconectare forțată).

Această abordare non-blocantă este esențială pentru fluiditatea jocului, permițând clientului să reacționeze instantaneu la mesajele serverului (ex: "Timpul a expirat", "Serverul a închis

conexiunea") chiar dacă utilizatorul nu interacționează cu tastatura în acel moment. De asemenea, clientul include logică de auto-terminare la detecția cuvintelor cheie de deconectare transmise de server.

3.3. Diagrama Detaliată a Aplicației

Fluxul de Joc (QuizzGame Multithreaded)



4. Protocolul de Comunicare

Comunicarea între componentele sistemului se realizează printr-un protocol la nivelul aplicație, orientat pe text (text-based), implementat peste stiva TCP/IP. Mesajele sunt delimitate prin caracterul de linie nouă (\n), iar interpretarea lor este dependentă de starea curentă a jocului (Context-Aware).

4.1. Structura Comenzilor (Client -> Server)

Comandă	**Parametri**	**Exemplu**	**Descriere**
LOGIN	<Nume>	LOGIN Ion	Solicită înregistrarea clientului în sesiunea de joc (Lobby). Serverul asociază numele cu socket-ul curent.
QUIT	-	QUIT	Anunță serverul că utilizatorul părăsește voluntar aplicația. Serverul închide socket-ul și eliberează resursele.
INPUT	<Text>	A, B, Paris Y, N	1. În timpul jocului: Reprezintă răspunsul la întrebarea curentă. 2. La final (Votare): Reprezintă opțiunea de restart (Y=Da, N=Nu).

4.2. Structura Răspunsurilor (Server -> Client)

Prefix / Tip	**Exemplu Mesaj**	**Descriere**
ACK_LOGIN	ACK_LOGIN: Salut Ion...	Confirmă autentificarea și informează clientul să aștepte startul jocului.

ÎNTREBARE	ÎNTREBAREA 1: ...	Mesaj broadcast care conține textul întrebării, variantele și timpul limită. Clientul îl afișează imediat.
STATUS	Răspuns înregistrat.	Confirmare individuală trimisă clientului după ce acesta a trimis un input valid.
REZULTAT	TIMP EXPIRAT! Corect: B	Anunță sfârșitul rundei curente și afișează clasamentul intermediar.
VOTARE	DORIȚI SA JUCAȚI DIN NOU?	Inițiază procedura de votare pentru restartarea serverului.
EXIT	Deconectare...	Mesaj critic: Instruiește clientul să-și încheie execuția locală (terminare proces) deoarece a fost eliminat sau jocul s-a încheiat.

5. Scenarii de utilizare

Pentru validarea arhitecturii, au fost testate următoarele fluxuri de execuție care acoperă atât funcționarea nominală, cât și situațiile de eroare:

5.1. Scenarii de Execuție cu Succes

1. **Fluxul Complet de Joc și Restartare (Ciclu Infinit):**
    - Doi clienți se conectează și se autentifică (LOGIN).
    - Serverul detectează atingerea pragului minim și pornește jocul.
    - După epuizarea întrebărilor și afișarea câștigătorului, serverul inițiază etapa de Votare.
    - Ambii clienți răspund cu Y (Da). Serverul îi menține conectați, resetează scorurile și îi transferă imediat în Lobby pentru o nouă rundă, fără a fi necesară reconectarea manuală.

2. **Sincronizarea cu Polling (Timp de Răspuns):**
    - Serverul trimite o întrebare cu limita de 10 secunde.
    - Clientul A răspunde rapid (secunda 2).
    - Serverul înregistrează răspunsul, dar nu trece imediat mai departe. În schimb, execută o buclă de verificare ("polling") secundă cu secundă până la expirarea timpului. Aceasta asigură că toți jucătorii au șanse egale, dar permite și detecția instantanee a eventualelor erori de rețea în timpul așteptării.

5.2. Scenarii de Eșec și Tratarea Erorilor

1. **Joc Anulat prin Lipsă de Jucători (Underflow):**
    - Într-un joc cu 2 participanți, Clientul A se deconectează forțat (ex: pană de curent/internet) în mijlocul unei runde.
    - Datorită mecanismului de polling, serverul detectează dispariția Clientului A în maxim 1 secundă.
    - Deoarece numărul de jucători activi scade sub pragul minim (2), serverul oprește imediat runda curentă, trimite un mesaj de notificare Clientului B ([!] Joc anulat...) și resetează starea serverului la "Lobby". Clientul B rămâne conectat și așteaptă un nou adversar.
2. **Eliminare prin Vot Negativ sau Inactivitate:**
    - La finalul jocului, în etapa de votare, un client răspunde cu N sau nu răspunde deloc în cele 15 secunde alocate.
    - Serverul îi trimite mesajul de încheiere (Deconectare...) și închide conexiunea TCP.
    - Clientul recepționează mesajul, detectează cuvântul cheie și își încheie execuția locală automat.

6. Concluzii

Soluția implementată pentru sistemul "QuizzGame" demonstrează viabilitatea unei arhitecturi concurente de tip **Client-Server Multithreaded** pentru aplicațiile care necesită sincronizare strictă în timp real. Utilizarea bibliotecii POSIX Threads a permis gestionarea eficientă a resurselor serverului, iar integrarea mecanismelor de *polling* a asigurat o robustețe ridicată la erori (Fault Tolerance), prevenind blocajele cauzate de deconectările imprevizibile ale utilizatorilor.

De asemenea, adoptarea modelului de multiplexare I/O (select) la nivelul clientului a îmbunătățit semnificativ experiența utilizatorului, permițând recepționarea notificărilor critice (stop joc, deconectare) instantaneu, independent de acțiunile utilizatorului la tastatură.

Potențiale îmbunătățiri: Deși aplicația îndeplinește toate specificațiile funcționale, aceasta poate fi extinsă prin:

- **Securitate (Criptare):** Implementarea unui strat de securitate **SSL/TLS** (folosind OpenSSL) peste conexiunea TCP pentru a cripta schimbul de mesaje, protejând astfel răspunsurile și datele utilizatorilor împotriva interceptării (Man-in-the-Middle).
- **Autentificare Persistentă:** Trecerea de la simpla identificare prin nume ("LOGIN Nume") la un sistem complet de înregistrare cu parolă. Parolele ar trebui stocate în baza de date SQLite sub formă de hash-uri (ex: SHA-256) pentru securitate.
- **Interfață Grafică (GUI):** Înlocuirea interfeței actuale în linie de comandă (CLI) cu o interfață grafică prietenoasă, dezvoltată folosind biblioteci precum Qt sau GTK, care ar permite afișarea mai atractivă a întrebărilor și a cronometrului.
- **Modul de Administrare:** Adăugarea unui tip special de client ("Admin") care să poată adăuga sau modifica întrebări în baza de date direct din aplicație, fără a opri serverul.

7. Referințe Bibliografice

- Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004). *UNIX Network Programming, Volume 1: The Sockets Networking API* (3rd ed.). Addison-Wesley Professional.
- Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer Networks* (5th ed.). Pearson.
- Butenhof, D. R. (1997). *Programming with POSIX Threads*. Addison-Wesley Professional.
- Kerrisk, M. (2010). *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press.
- Postel, J. (1981). *Transmission Control Protocol*. RFC 793. Internet Engineering Task Force. Disponibil la: https://tools.ietf.org/html/rfc793
- Hipp, D. R., & Kennedy, D. (2024). *SQLite Documentation*. Disponibil la: https://www.sqlite.org/docs.html
- Linux Programmer's Manual (man pages): socket(2), bind(2), listen(2), accept(2), select(2), pthread_create(3), pthread_mutex_lock(3).