VirtualSoc - Rețea Socială Distribuită cu Grupuri, Prieteni, Mesagerie și Notificări Persistente

Toma Ionut-Cristian

Facultatea de Informatică, Universitatea „Alexandru Ioan Cuza”, Iași, România
ionut-cristian.toma@info.uaic.ro

Abstract *VirtualSoc* este o platformă software de tip rețea socială, construită pe o arhitectură client–server, ce permite utilizatorilor să posteze, să interacționeze prin mesaje private și de grup, să gestioneze relații de prietenie și să creeze grupuri tematice. Sistemul implementează un protocol text custom peste TCP, persistă datele cu SQLite3 și asigură procesarea concurentă a conexiunilor prin multithreading. În versiunea actualizată, clientul utilizează multiplexare I/O prin `select()` pentru a primi asincron răspunsuri și notificări, iar notificările sunt persistate în baza de date și pot fi consultate/șterse prin comenzi dedicate. Pentru răspunsuri mari (feed, liste), serverul transmite în bucăți și delimitează fluxul cu marcatorul `END`.

Keywords: C, Multithreading, Client-Server, TCP, SQLite, Protocol Custom, select(), Notifications

1 Introducere

VirtualSoc este o rețea socială academică, dezvoltată ca aplicație client–server, ce oferă funcționalități de bază pentru interacțiunea utilizatorilor: postări, mesagerie privată, grupuri, gestionarea prieteniilor și vizibilitate controlată a conținutului. Proiectul pune accent pe controlul datelor, modularitate și extensibilitate, fiind o alternativă self-hosted la platformele comerciale.

Obiectivele tehnice principale sunt:

1. **Protocol de Comunicare Custom:** Definirea unui protocol text simplu pentru comenzi și răspunsuri, cu parsare robustă și validare a argumentelor.
2. **Persistență și Consistență:** Utilizarea SQLite3 pentru stocarea utilizatorilor, postărilor, relațiilor, mesajelor, grupurilor și a notificărilor persistente.
3. **Concurență:** Serverul gestionează conexiuni multiple simultan folosind `pthreads`, cu sincronizare pe accesul la baza de date.
4. **Controlul Vizibilității:** Implementarea nivelurilor de vizibilitate pentru postări (public, friends, close friends) și reguli clare de acces.
5. **Notificări Asincrone:** Livrarea evenimentelor (ex: DM, cereri prietenie, mesaje de grup) în timp real către clienți conectați și persistarea lor pentru consultare ulterioară.

6. **Robustețe la mesaje mari:** Trimiterea răspunsurilor voluminoase în segmente și delimitarea completării cu `END`, evitând depășirea bufferelor pe client.

2 Tehnologii Aplicate

- **C (C11):** Limbajul principal pentru server și client, ales pentru controlul resurselor, performanță și portabilitate. Codul este structurat pe module (autentificare, postări, prieteni, grupuri, mesaje, notificări).
- **Protocol TCP/IP:** Transport fiabil pentru comenzi și răspunsuri. Aplicația folosește socket-uri TCP și I/O blocking, iar clientul multiplexează sursele de input cu `select()`.
- **select():** Pe client, `select()` permite așteptarea simultană pe `STDIN` și pe socket, astfel încât notificările să apară imediat fără blocarea UI-ului.
- **SQLite3:** Bază de date embedded pentru persistența entităților (utilizatori, postări, mesaje, prietenii, grupuri, membri, cereri) și **notificări**. Se folosesc statement-uri `prepare/bind/step` și un mutex global pe accesul concurent.
- **POSIX Threads (pthreads):** Serverul tratează fiecare client într-un thread dedicat. Pentru consistență, accesul la `g_db` este protejat cu mutex.
- **Libsodium:** Hasharea securizată a parolelor (ex: Argon2), protejând împotriva atacurilor brute-force și a stocării parolelor în clar.
- **Makefile:** Automatizează build-ul și linkarea dependențelor (`-lsqlite3 -lsodium -lpthread`).

3 Structura Aplicației

3.1 Componenta Server (`server_app`)

Serverul ascultă conexiuni pe un port și procesează fiecare client într-un thread separat. Module principale:

1. **Main Loop:** Acceptă conexiuni și lansează thread-uri per client.
2. **Command Dispatcher (`command_dispatch.c`):** Citește comenzi text, parsează `cmd/arg1/arg2`, validează autentificarea și apelează logica de business.
3. **Database Layer / Storage:** Operații SQLite pentru toate entitățile. Accesul este serializat cu mutex global.
4. **Business Logic:**
   - **Postări:** creare, feed public/personalizat, ștergere, formatare și trimitere în bucăți (`posts_send_for_client`).
   - **Mesaje (DM):** conversații, istoric, trimitere, notificare către destinatar.
   - **Grupuri:** creare, join/request, approve/reject, membri, mesagerie de grup, notificări către membrii grupului.
   - **Prieteni (model actualizat):** `add` devine **cerere** către țintă; ținta poate vedea cererile (`view_friend_requests`) și poate accepta/respinge; în plus, în contextul cererilor, `add` pentru țintă poate acționa ca **accept** (aceeași operație logică).

- **Notificări:** la evenimente, serverul (1) persistă notificarea în DB și (2) dacă utilizatorul este online, o trimite imediat prin `notify_user`.

5. **Sesiuni (`sessions.c`):** mapare `user_id -> client_fd` (ex: `sessions_find_fd_by_user_id`) pentru livrarea notificărilor în timp real.

3.2 Componenta Client (`client_app`)

Clientul oferă o interfață CLI:

- **UI CLI:** comenzi, prompt, colorare (OK verde, INFO/NOTIF galben, ERROR roșu).
- **Multiplexare cu `select()`:** clientul așteaptă simultan input de la tastatură și date de la server. Notificările sunt afișate imediat și promptul este reafișat (fără a necesita apăsarea Enter).
- **Receiver robust pentru răspunsuri mari:** pentru feed/liste/istoric, clientul acumulează date și procesează linii până la `END`. Acest lucru evită truncarea și depășirea bufferelor.
- **Protocol Client (`protocol_client.c`):** serializează comenzi (o linie per cerere) și le trimite către server.

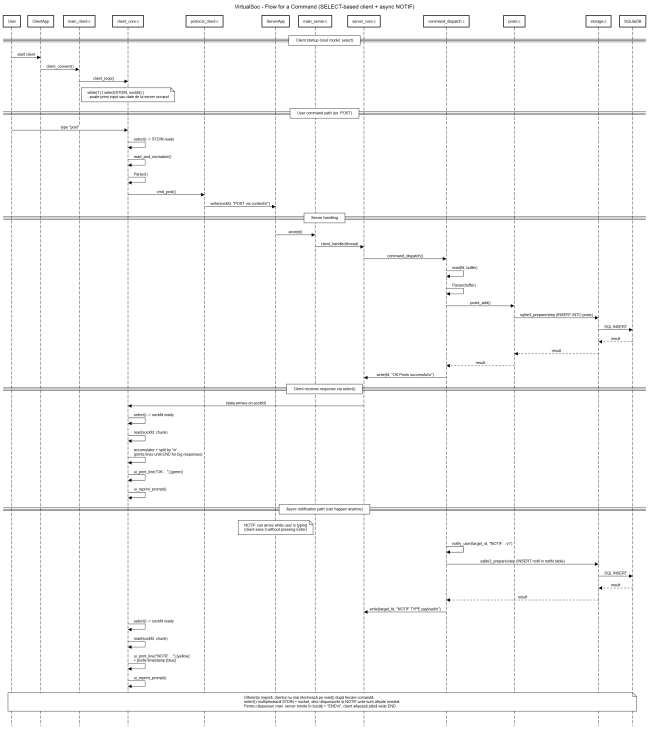

Figura 1. Arhitectura aplicației VirtualSoc: fluxul comenzilor, multiplexare cu `select()` și notificări

4 Protocolul de Comunicare

Protocolul este text-based, fiecare cerere este o linie terminată cu `\n`. Serverul răspunde cu linii prefixate `OK`, `ERROR`, `INFO` sau `NOTIF`. Pentru răspunsuri voluminoase, serverul transmite mai multe linii și termină fluxul cu linia `END`.

4.1 Convenții generale

- **Cerere:** `CMD [arg1] [arg2...]\n`
- **Răspuns scurt:** o singură linie `OK ...` sau `ERROR CODE ...`
- **Răspuns lung:** antet (ex: `OK`), apoi payload pe mai multe linii, terminator `END`
- **Notificare:** `NOTIF <TYPE> <payload>\n` (livrată asincron către utilizatorul țintă)

4.2 Comenzi principale (exemple)

- `REGISTER <user> <pass>`
- `LOGIN <user> <pass>`
- `LOGOUT`
- `POST <public|friends|close> <content...>`
- `VIEW_PUBLIC_POSTS`
- `VIEW_FEED`
- `VIEW_USER_POSTS <user>`
- `SEND_MESSAGE <user> <text...>`
- `LIST_MESSAGES <user>`
- `ADD_FRIEND <user>` (creează cerere; în cazul țintei poate funcționa ca accept)
- `VIEW_FRIEND_REQUESTS`
- `REJECT_FRIEND <user>` (respinge cererea)
- `LIST_FRIENDS`
- `CREATE_GROUP <name> <PUBLIC|PRIVATE>`
- `JOIN_GROUP <name>`
- `REQUEST_GROUP <name>`
- `APPROVE_GROUP_MEMBER <group> <user>`
- `REJECT_GROUP_REQUEST <group> <user>`
- `SEND_GROUP_MSG <group> <text...>`
- `GROUP_MESSAGES <group>`
- `VIEW_NOTIFS`
- `DELETE_NOTIFS`

4.3 Tipuri de notificări (exemple)

- `NOTIF DM_FROM <sender> <text...>`
- `NOTIF FRIEND_REQUEST <sender>`
- `NOTIF GROUP_MSG <group> <sender> <text...>`

5 Scenarii de Utilizare

5.1 Execuție cu Succes

1. **Mesaj privat cu notificare în timp real:** Alice trimite `SEND_MESSAGE Bob "Salut!"`. Serverul persistă mesajul și trimite către Bob o notificare `NOTIF DM_FROM ...`. Clientul lui Bob o afișează imediat datorită `select()`, fără a întrerupe interacțiunea.
2. **Feed personalizat (răspuns mare):** Utilizatorul rulează `VIEW_FEED`. Serverul trimite antetul `OK`, apoi postări formate pe mai multe linii și termină cu `END`. Clientul acumulează și afișează liniile până la `END`, evitând limitările unui singur buffer.
3. **Notificări persistente:** Un utilizator offline primește evenimente (ex: cerere prietenie). Serverul salvează notificările în DB. La reconectare, utilizatorul rulează `VIEW_NOTIFS` și vede toate notificările nesterse; apoi rulează `DELETE_NOTIFS`, iar o nouă interogare `VIEW_NOTIFS` nu mai afișează notificările vechi.

5.2 Execuție cu Eșec

1. **Autentificare eșuată:** Utilizatorul trimite `LOGIN` cu parolă greșită. Serverul returnează `ERROR ERR_... Login failed`, iar clientul afișează mesajul în roșu.
2. **Acces nepermis la conținut:** Un utilizator încearcă `VIEW_FEED` sau `VIEW_USER_POSTS` fără permisiunile corespunzătoare (profil privat, fără prietenie mutuală). Serverul răspunde cu `ERROR ERR_NO_PERMISSION ...`.
3. **Comandă invalidă / argumente lipsă:** Clientul trimite o comandă fără argumentele necesare (ex: `SEND_GROUP_MSG` fără text). Serverul răspunde `ERROR ERR_BAD_ARGS Usage: ...`, iar comanda nu este executată.

6 Concluzii

VirtualSoc demonstrează implementarea unei rețele sociale distribuite cu funcționalități de bază și un set extins de mecanisme pentru robusteză și UX: multiplexare cu `select()` pe client, notificări asincrone, persistența notificărilor și suport pentru răspunsuri mari delimitate prin `END`. Arhitectura modulară și protocolul text permit extinderea facilă.

Direcții Viitoare

- **Separarea canalelor:** un canal dedicat pentru notificări (de ex. al doilea socket) sau includerea unui `message-id` pentru a diferenția răspunsuri vs. notificări mai robust.
- **TLS/SSL:** securizarea comunicației TCP.
- **Rate limiting și anti-spam:** limitarea notificărilor și mesajelor abuzive.
- **Optimizare DB:** indici pentru feed și notificări, tranzacții pentru operații compuse.
- **Interfață GUI:** client cu UI grafic.

7 Referințe Bibliografice

Bibliografie

1. Springer: *LNCS Conference Proceedings Guidelines*. https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines
2. Stevens, W. R., Fenner, B., Rudoff, A. M.: *UNIX Network Programming, Volume 1: The Sockets Networking API*. Addison-Wesley (2003)
3. The Open Group: *select() — POSIX.1 Specification*. https://pubs.opengroup.org/onlinepubs/9699919799/functions/select.html
4. The Open Group: *POSIX Threads (pthreads)*. https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html
5. SQLite: *SQLite Documentation*. https://www.sqlite.org/docs.html
6. libsodium: *Documentation*. https://doc.libsodium.org/