SimplePubSubTopicBroker

Alexandru Butnaru, 2B4
Facultatea de Informatică, Universitatea Alexandru Ioan Cuza, Iaşi

Decembrie 2023

Cuprins

1 Introducere

1.1 Viziunea generală

Dată fiind cantitatea uriașă de informații publicate pe Internet în fiecare zi, unei persoane i-ar fi imposibil s-o asimileze pe toată și, de asemenea, nu-și dorește să-i fie trimis fiecare articol nou adăugat.
Aplicația **SimplePubSubTopicBroker** are ca scop furnizarea de articole special selectate să se potrivească preferințelor cititorului, precum și un mediu pentru scriitori de a-și publica articolele.

1.2 Obiective

- Crearea unei baze de date bine structurată pentru stocarea unor scurte publicații (articole).
- Realizarea unei aplicații de gestionare eficientă a acesteia, în contextul utilizării sale de către mai mulți clienți concomitent.
- Categorizarea articolelor în funcție de topic, pentru a asigura trimiterea către abonați doar a subiectelor de interes.
- Implementarea unui sistem de înregistrare & conectare.
- Eliminarea redundanței: un sistem care trimite abonaților, la cerere, doar articolele noi, necitite.

2 Tehnologii Aplicate

2.1 TCP/IP

Transmission Control Protocol / Internet Protocol este un ansamblu de protocoale de comunicare între dispozitivele dintr-o rețea, bazat pe patru nivele: Aplicație, Transport, Rețea, și Conexiune. Am ales acest protocol deoarece, spre deosebire de **UDP**, oferă comunicare în ambele sensuri (**full-duplex**), securitate (**"three-way handshake"**), precum și transmiterea corectă și completă a pachetelor de date.

2.2 SQLite

SQLite este o librărie open-source pentru limbajul **C** creată pentru a oferi posibilitatea integrării bazelor de date în aplicații. Utilizarea acestei librării este justificată de următoarele avantaje pe care le prezintă: suportul pentru baze de date de dimensiuni foarte mari, timpul scăzut de răspuns (baza de date fiind stocată local, nu este necesară conectarea la un server de găzduire), și interfața intuitivă (în afară de cunoștințe legate de baze de date, un minim de efort este necesar pentru a folosi funcțiile librăriei).

3 Structura Aplicației

3.1 Diagrama generală a aplicației

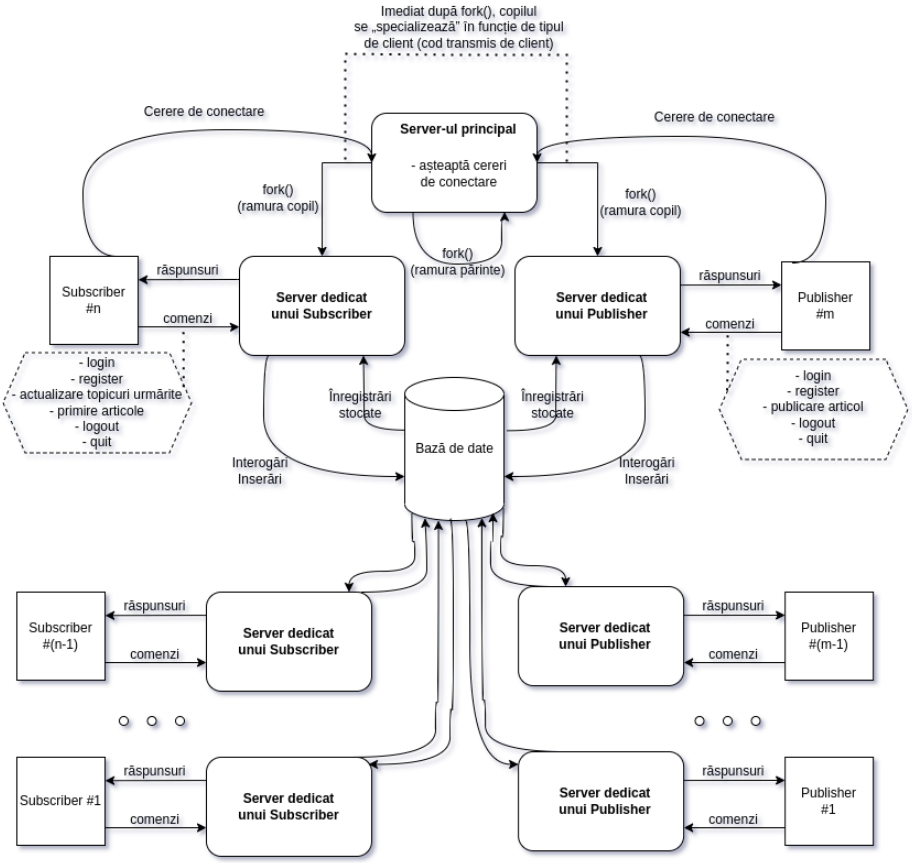

Figure 1: Diagrama de funcționare a aplicației

3.2 Concepte implicate

- Serverul este unul de tip **TCP/IP Concurent**: la apariția unei cereri de conectare din partea unui client, serverul se va duplica folosind primitiva **fork()**, din standardul **POSIX** pentru limbajul C. Astfel, procesul copil creat va servi clientul nou-conectat, pe când procesul părinte va rămâne în starea de așteptare de noi conexiuni, permițând astfel servirea concomitentă a unui număr teoretic (presupunând un sistem cu resurse fizice inepuizabile) nelimitat de clienți.

- Conceptul de **control al accesului** la baza de date este dat de mecanismul de conectare cu parolă: scrierile în și citirile din baza de date sunt realizate doar de către instanțele serverului, care refuză orice operațiune de acest tip venită din partea unui utilizator neconectat.

- Mecanismul de **evitare a fenomenelor de tip data-race** de la baza aplicației: serverul deschide baza de date pentru citire sau scriere exact înaintea acestor operații și o închide imediat după, asigurând astfel că se va ține cont de intrările care au fost adăugate de către alt client paralel între timp.

4 Aspecte de Implementare

4.1 Baza de date

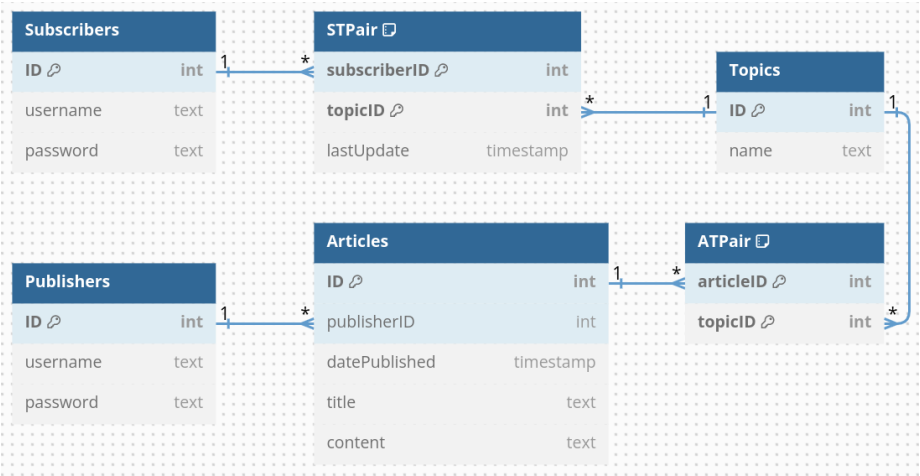

Figure 2: Structura bazei de date

Din moment ce un cititor poate fi interesat de mai multe subiecte, iar un articol poate fi încadrat în mai multe topicuri, apar relații many-to-many între tabela **Topics** și **Subscribers**, respectiv **Articles**. Tabelele intermediare **STPair** și **ATPair** asigură că există doar relații one-to-many în baza de date. Coloana "lastUpdate" a fiecărei pereche **Subscriber-Topic** permite, la cererea acestuia, să primească doar acele articole pe care nu le-a citit încă legate de acel subiect, pentru o experiență-utilizator cât mai fluidă.

4.2 Protocolul la nivel aplicație

La nivel de aplicație, protocolul folosit poate fi considerat ca **Simple Mail Transfer Protocol**: mesajele (articolele) sunt reprezentate de text ASCII simplu, se realizează trimiterea a unul sau mai multe mesaje, mesajele sunt stocate pe un server și pot fi reaccesate ulterior, iar transmiterea este asincronă (cititorul nu trebuie să fie conectat în momentul publicării articolului).
Datorită existenței broker-ului (scriitorii netrimițând articolele direct cititorilor), aplicația dezvoltată se încadrează în tiparul **Publish-Subscribe**, varianta topic-based.

4.3 Secvențe de cod relevante

```
client = accept(socketDescriptor, (struct sockaddr *)&communicationSocket, &length);
if (client < 0){
    perror("[MasterBroker] Eroare la accept().\n");
    continue;
}
int pid;
if ((pid = fork()) == -1){
    printf("[MasterBroker] Eroare la fork().\n");
    close(client);
    continue;
}
else if (pid > 0){
    close(client);
    while (waitpid(-1, NULL, WNOHANG));
    continue;
}
else if (pid == 0){
    close(socketDescriptor);
    if (read(client, &controlCode, sizeof(int)) <= 0){
        printf("[Broker#%d] Eroare la read() de la client.\n", getpid());
        close(client);
        continue;
    }
    switch (controlCode){
    case 1:
        printf("[Broker#%d] Acum servesc un Subscriber.\n", getpid());
        serverType = 's';
        serveSubscriber(client);
        break;
    case 2:
        printf("[Broker#%d] Acum servesc un Publisher.\n", getpid());
        serverType = 'p';
        servePublisher(client);
        break;
```

Figure 3: Acceptarea unui client, clonarea și "specializarea" server-ului

Serverul va accepta conexiuni solicitate la orice adresă local-host (127.0.0.x, adrese de tipul IPv4, datorită parametrului **AF_INET** dat apelului socket()), și își va rezerva portul 2024 (predefinit).
Imediat după conectarea la server, clientul își specifică tipul (scriitor/cititor) printr-un cod, procesul-copil al server-ului știind astfel ce funcționalități să-i permită. Procesul-părinte revine la așteptarea de noi clienți.

```c
p = strtok(topics, ",");
while (p != NULL)
{
    tCount++;
    bzero(response, MAXMSGLEN);
    sprintf(command, "SELECT ID FROM Topics WHERE name = lower('%s');", p);
    sqlite3_prepare_v2(database, command, -1, &statement, NULL);
    if (sqlite3_step(statement) == SQLITE_ROW)
        topicID = sqlite3_column_int(statement, 0);
    else
    {
        sqlite3_finalize(statement);
        sprintf(command, "INSERT INTO Topics (name) VALUES (lower('%s'));", p);
        sqlite3_prepare_v2(database, command, -1, &statement, NULL);
        if (sqlite3_step(statement) != SQLITE_DONE)
        {
            sqlite3_finalize(statement);
            sprintf(response, "!Nu am putut crea topicul \"%s\".", p);
            sendResponse(client, response);
            p = strtok(NULL, ",");
            continue;
        }
        topicID = sqlite3_last_insert_rowid(database);
    }

    sqlite3_finalize(statement);
    sprintf(command, "INSERT INTO ATPair (articleID, topicID) VALUES (%d, %d);", articleID, topicID);
    sqlite3_prepare_v2(database, command, -1, &statement, NULL);
    if (sqlite3_step(statement) == SQLITE_DONE)
        sprintf(response, "!Am legat articolul de topicul \"%s\".", p);
    else
        sprintf(response, "!Nu am putut lega articolul de topicul \"%s\".", p);
    sqlite3_finalize(statement);

    sendResponse(client, response);
    p = strtok(NULL, ",");
```

Figure 4: Varianta *Topic*: Atribuirea de topic-uri unui articol

Tabelul **ATPair** face legătura între articole și topic-urile asociate. Dacă un topic menționat de scriitor nu există, atunci este creat mai întâi.

```c
sprintf(command, "SELECT DISTINCT a.title, p.username, a.content "
                 "FROM Publishers p "
                 "JOIN Articles a ON p.ID = a.publisherID "
                 "JOIN ATPair atp ON a.ID = atp.articleID "
                 "WHERE a.datePublished > '%s' COLLATE NOCASE AND atp.topicID IN "
                 "  (SELECT topicID "
                 "   FROM STPair "
                 "   WHERE subscriberID = %d);",
        lastUpdateTimestamp, currentUser);
sqlite3_prepare_v2(database, command, -1, &statement, NULL);
while (sqlite3_step(statement) == SQLITE_ROW)
{
    sprintf(response, "!\t\"%s\" - de '%s'\n  %s",
            sqlite3_column_text(statement, 0),
            sqlite3_column_text(statement, 1),
            sqlite3_column_text(statement, 2));
    sendResponse(client, response);
    artCount++;
}
sqlite3_finalize(statement);
sprintf(response, ".Total: %d articol(e).", artCount);
sendResponse(client, response);

sprintf(command, "UPDATE STPair "
                 "SET lastUpdate = '%s' "
                 "WHERE subscriberID = %d;",
        currentTimestamp, currentUser);
sqlite3_prepare_v2(database, command, -1, &statement, NULL);
if (sqlite3_step(statement) != SQLITE_DONE)
    printf("[Broker#%d] Eroare la actualizat data ultimului update: %s.\n", getpid(), sqlite3_errmsg(database));
sqlite3_finalize(statement);
```

Figure 5: Varianta *Topic*: Livrarea articolelor noi de interes unui cititor

Server-ul preia din baza de date doar acele articole care:
1) au asociate cel puțin un topic din lista de interese al cititorului curent;
2) au data publicării ulterioară datei ultimei verificări pentru topicul de legătură.

Mesajele care încep cu "**!**" anunță clientul să aștepte mai multe mesaje de la server, iar "**.**" îi spune că acesta este ultimul și se poate întoarce la interfața de comenzi. Astfel, se asigură livrarea tuturor mesajelor într-o manieră formatată, fără pierderi/erori din cazua buffer-ului **response** de dimensiune fixă.

Livrarea tuturor articolelor noi este urmată în server de actualizarea câmpurilor "lastUpdate" cu data & ora curente.

4.4 Scenariu de utilizare

Cititor1 creează cont și se abonează la subiectele "A" și "B", apoi solicită toate articolele existente.
Broker-ul caută și trimite la Cititor1 toate articolele etichetate cu aceste topicuri.
Cititor1 se deconectează.
Între timp, Scriitor5 adaugă un articol nou, "Titlu" în categoriile "A" și "C".
Cititor1 se conectează, solicită doar articolele noi, și primește de la broker "Titlu".
Cititor2 se conectează, cere articolele noi și, datorită preferințelor, nu primește niciunul. Adaugă "C" în lista topicurilor și, cerând iarăși articolele noi, primește articolul "Titlu" (și posibil altele mai vechi).

5 Concluzii

Deși aplicația dezvoltată este un sistem ușor de folosit pentru crearea și distribuirea de articole cititorilor în funcție de interesele acestora, există loc de îmbunătățiri:

- la un număr foarte mare de utilizatori, metoda de evitare a data-race-urilor devine ineficientă; soluțiile posibile includ utilizarea de **lacăte în citire/scriere** asupra bazei de date, sau limitarea accesului la aceasta prin **semafoare**;
- **criptarea** parolelor;
- compatibitate cu un al treilea tip de client: **Moderator** - cu permisiunea de a șterge articole, dacă acestea sunt spam, false, sau jignitoare.

Alte aspecte necritice de îmbunătățire sunt: funcții de rating și comentarii pentru articole, căutarea unui anumit articol, notificarea cititorilor la apariția unui nou articol de interes fără necesitatea cererii explicite.

6 Referințe Bibliografice

[1] https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php

[2] https://ro.wikipedia.org/wiki/TCP/IP

[3] https://www.geeksforgeeks.org/tcp-3-way-handshake-process/

[4] https://www.sqlite.org/index.html

[5] https://ro.wikipedia.org/wiki/SMTP

[6] https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern