ACADEMIA TEHNICĂ MILITARĂ “FERDINAND I”

FACULTATEA DE SISTEME INFORMATICE ŞI SECURITATE CIBERNETICĂ

Specializarea: Calculatoare şi sisteme informatice pentru apărare şi securitate naţională

Proiectarea Sistemelor de Operare-Proiect
Message Queues

STIR Catalin Ionut

AIONESEI Claudiu Marian

C-113A

Cuprins

1. Introducere

1.1 Scopul proiectului

Proiectul isi propune sa implementeze un sistem de cozi de mesaje care sa permita schimbul de informatii intre mai multi participanti intr-un mod asincron. Acest sistem de mesagerie este similar cu modelul "publish-subscribe", in care mesajele sunt transmise de catre "publishers" prin canale dedicate, iar "subscribers" primesc doar mesajele de interes prin abonarea la anumite tipuri de mesaje sau chiar cu ajutorul unor filtre pe baza continutului mesajelor.

Un astfel de sistem de cozi de mesaje este util in dezvoltarea de aplicatii distribuite sau de procesare a datelor in timp real, unde comunicarea intre componente trebuie sa fie eficienta si sa suporte o scalare flexibila.

1.2 Componente principale

In dezvoltarea proiectului ne vom utiliza de urmatoarele mari componente:

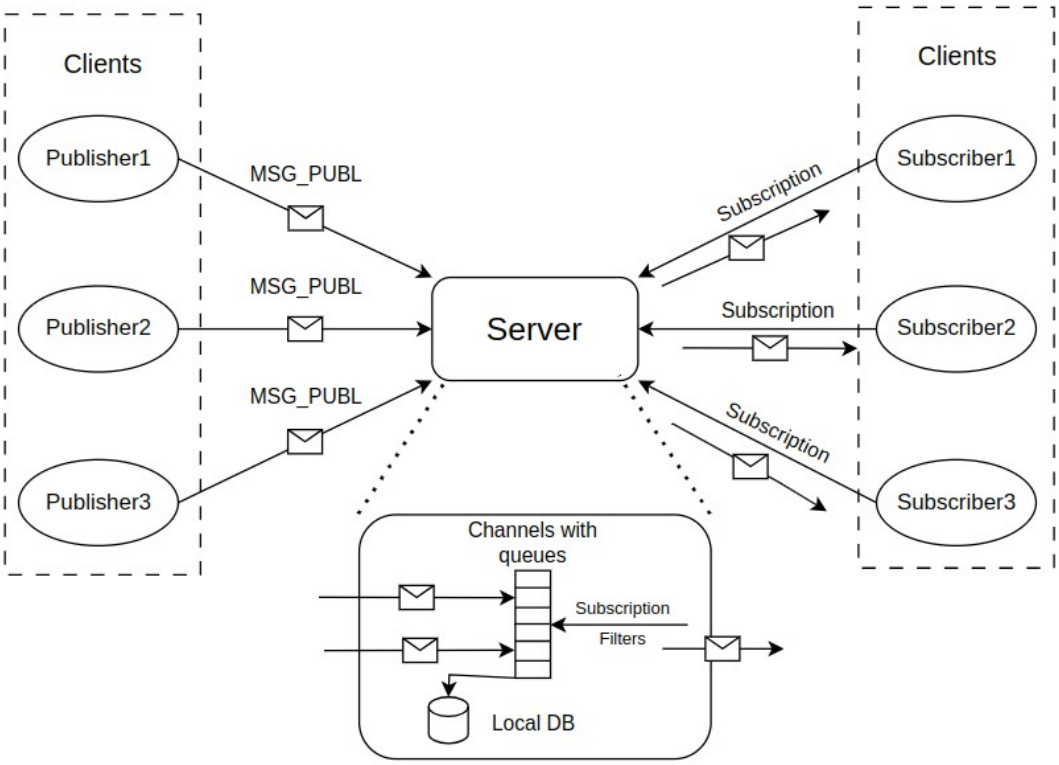
1. **Publisher** - Un proces client care genereaza mesaje si le publica pe un canal specific. Fiecare publisher va crea un canal de comunicare propriu, canal format dintr-o coada de mesaje, pentru mesajele pe care le va trimite.

2. **Server** - Componenta centrala care intermediaza comunicarea intre publisheri si subscriberi. Serverul primeste mesajele de la publisheri pe care le asociaza unor canale si le distribuie catre subscriberi in functie de tipul de mesaj la care acestia sunt abonati.

3. **Subscriber** - Un proces care se aboneaza la unul sau mai multe tipuri de mesaje si primeste mesaje legate de starea abonamentelor sau a serverului de la server.

1.3 Diagrama

Diagrama ce ilustreaza workflow-ul aplicatiei si componentele ei:



2. Detalierea Cerintelor si Componentelor

2.1 Cerinte Functionale

1. **Crearea de Canale de Comunicare pentru Publisheri**:
   - Fiecare publisher trebuie sa poata crea unul sau mai multe canale dedicate pentru transmiterea mesajelor.
   - Publisherii trebuie sa poata adauga noi mesaje in canalele proprii, care vor fi gestionate de server.
   - Canalele de comunicare trebuie sa suporte mai multi subscriberi.

2. **Intermedierea mesajelor**:
   - Serverul va gestiona cozile de mesaje cu ajutorul unor metode de eficientizare, folosind-use de niste structuri de date si algoritmi potriviti si va sustine mecanismul de publish-subscribe.
   - Acesta trebuie sa pastreze o evidenta a canalelor si a subscriberilor abonati la fiecare canal.
   - La primirea unui mesaj, serverul trebuie sa distribuie mesajul doar catre subscriberii care sunt abonati la canalul respectiv (tinand cont si de posibilele filtre ale subscriberilor).

3. **Abonarea Subscriberilor**:

- Subscriberii se pot abona la unul sau mai multe canale si primesc doar mesajele de pe aceste canale.
- Fiecare subscriber va putea filtra mesajele in functie de un keyword la alegerea acestuia.

4. **Asigurarea Sincronizarii si a Concurentei**:
   - Sistemul trebuie sa permita mai multi publisheri si subscriberi sa lucreze simultan, fara a crea blocaje.
   - Cozile de mesaje vor fi gestionate sincron pentru a evita conflictele de acces atunci cand mai multi subscriberi citesc simultan.

2.2 Cerinte Tehnice

- **Limbajul de Programare:** Implementarea va fi realizata in limbajul *C* compilat in *gcc* folosind utilitarul *make*.
- **Biblioteci:** Se vor utiliza functionalitati din biblioteca Standard C si biblioteca *json-c*.
- **Structura Mesajelor:** Fiecare mesaj va contine un header cu tipul mesajului, lungimea lui, topicul, subtopicul si payload-ul. Serverul va utiliza o structura de date cu abonamente, care are o cautare in complexitate O(log(n)) pentru a redirectiona mesajele corespunzator.

3. Descrierea Componentelor

3..1. Publisher:

- **Crearea Canalelor:** Fiecare publisher are posibilitatea de a trimite unul sau mai multe mesaje pe topicuri si subtopicuri diferite. De fiecare data cand se trimite un mesaj cu un nou topic si subtopic, se creeaza un nou canal de comunicare la care se pot abona subscriberii.

- **Transmiterea Mesajelor:** Publisherii transmit mesajele prin socketuri catre server. Mesajele vor fi stocate in niste structuri de date care identifica topicul mesajului pe baza unui Hash Table, iar subtopicul pe baza unui RBTree.

- **Implementarea Cozilor de Mesaje:** Publisherii nu utilizeaza un API de mesagerie preexistent, ci apeleaza la o coada de mesaje construita intern, care functioneaza pe baza socketurilor si a unui mecanism de notificare asincron (prin epoll). Acest mecanism gestioneaza eficient mesajele publicate si asigura distribuirea lor catre server.

3..2. Server:

- **Gestionarea Canalelor si Cozilor de Mesaje:** Serverul gestioneaza toate canalele deschise de publisheri si pastreaza evidenta subscriberilor abonati la fiecare canal. Serverul va gestiona fiecare canal folosind un model de cozi interne, in interiorul structurilor specificate mai devreme, asigurand transmiterea mesajelor in ordinea publicarii.

- **Procesarea si Distribuirea Mesajelor:** La primirea mesajelor de la publisheri prin socketuri, serverul foloseste epoll pentru monitorizarea conexiunilor si recv pentru a prelua mesajele din cozi. Mesajele sunt apoi analizate pentru a identifica daca e de tip mesaj sau de tip request(de la subscriber).

- **Notificarea Subscriberilor:** Serverul distribuie mesajele catre subscriberi doar daca acestia sunt abonati la canalul corespunzator si daca continutul mesajului este potrivit filtrului subscriberului (optional). Notificarile sunt transmise in mod asincron, utilizand socketuri si mecanisme de notificare eficienta.

3..3. Subscriber:

- **Inregistrarea si Abonarea la Canale:** Subscriberii se conecteaza la server si se aboneaza la unul sau mai multe canale. Acestia specifica canalele dorite pe baza unui topic si subtopic, iar serverul pastreaza o structura asemanatoare cu abonamentele fiecarui subscriber.

- **Consumul Mesajelor:** Subscriberii primesc mesajele corespunzatoare prin socketuri si le citesc folosind functia recv. Sistemul de mesagerie permite consumul asincron al mesajelor.

- **Gestionarea Mesajelor Primite:** Fiecare subscriber poate filtra mesajele pe baza tipului sau a altor parametri din JSON. Mesajele sunt procesate si stocate local in functie de cerintele aplicatiei.

4. Exemplu de workflow

- **Pasul 1:** Un publisher creeaza un mesaj si il adauga intr-o coada dedicata.

- **Pasul 2:** Serverul preia mesajul din coada publisherului, il adauga canalului aferent sau creaza unul daca este cazul.

- **Pasul 3: I**dentifica subscriberii abonati si le distribuie mesajul.

- **Pasul 4:** Subscriberii primesc mesajele si le proceseaza conform specificatiilor.