SmartParkingAssistant

Radu Dan-Ștefan

1. Introducere

Viziunea generala:
Aplicația va folosi un server central și mai mulți clienți distribuiți pentru a monitoriza locurile de parcare dintr-o parcare supraetajată. Fiecare client va gestiona un etaj specific, detectând în timp real locurile ocupate și libere, folosind senzori montați deasupra fiecărui loc de parcare care la randul lor vor fii tot clienți ce vor răspunde clientului responsabil pe tot etajul, raportand apoi informațiile către server. Serverul va centraliza datele și va oferi utilizatorilor o interfață pentru a vedea disponibilitatea locurilor. Asadar, exista 2 tipuri de clienti: cei ce au în subordine un etaj întreg și cei ce se adresează unui singur senzor.

Obiectivele aplicației

1.1. Monitorizare în timp real: Identificarea și raportarea rapidă a stării fiecărui loc de parcare folosind senzori de proximitate montați deasupra fiecărui loc de parcare.

1.2. Centralizare: Agregarea informațiilor despre toate etajele într-un server central.

1.3. Scalabilitate: Posibilitatea de a adăuga noi etaje sau locuri de parcare fără a afecta performanța sistemului.

1.4. Prezentarea informațiilor într-o aplicație user friendly.

2. Tehnologii Aplicate:

Comunicarea este realizată prin intermediul tehnologiei TCP/IP pentru a asigura o conexiune individuală cu fiecare client în parte. Nu este necesara metoda de comunicare UDP întrucât serverul nu are nimic de comunicat cu toți clienții conectați în același timp, sau cu alte instanțe neasociate.
Clienții ce au în subordine tot etajul vor primi de la fiecare senzor(alți clienți) modificări de stare a locului de parcare urmând ca printr-un socket de tip epoll urmând ca clienții ce au în subordine toată parcarea să comunice modificările către serverul central. Protocolul de comunicare stabilit este IPv4 întrucât functionalitatea avansată a protocolului IPv6 nu este necesară în această situație, programul fiind unul ce se concentrează pe simplitate și inteligibilitate. Totodată, nu este necesară encriptia pe care IPv6 o oferă. Acest protocol ar putea fi totuși implementat în viitor dată fiind modularitatea codului. Clienții comunica cu serverul (si intre ei) prin intermediul funcției EPOLL întrucât aceasta este necesară pentru a menține programul eficient și rapid când un număr mare de clienți comunică concurent.

3. Structura Aplicației

1. Sockets:

- **Socket-uri TCP:** Aplicația folosește socket-uri TCP pentru a crea o conexiune persistentă între client și server, asigurând o comunicare fiabilă și ordonată.
- **`socket()`:** Această funcție creează un nou socket. Argumentele specifică familia de adrese (IPv4 în acest caz), tipul socket-ului (STREAM pentru TCP) și protocolul (IPPROTO_TCP).
- **`bind()`:** Leagă socketul de o adresă IP și un port specific, permițând clienților să se conecteze la server.
- **`listen()`:** Pune socket-ul în modul de ascultare, așteptând conexiuni de la clienți.
- **`accept()`:** Acceptă o conexiune de la un client, creând un nou socket pentru comunicare.
- **`send()`/`recv()`:** Trimite și primește date prin socket.

2. I/O Multiplexing cu Epoll:

- **Epoll:** O metodă eficientă de a gestiona mai multe conexiuni simultan, permițând serverului să detecteze evenimente (cum ar fi datele primite) pe mai multe socket-uri fără a bloca firul de execuție.
- **`epoll_create1()`:** Creează o instanță epoll.
- **`epoll_ctl()`:** Adaugă, modifică sau elimină socket-uri din setul monitorizat de epoll.
- **`epoll_wait()`:** Așteaptă evenimente pe socket-urile monitorizate.

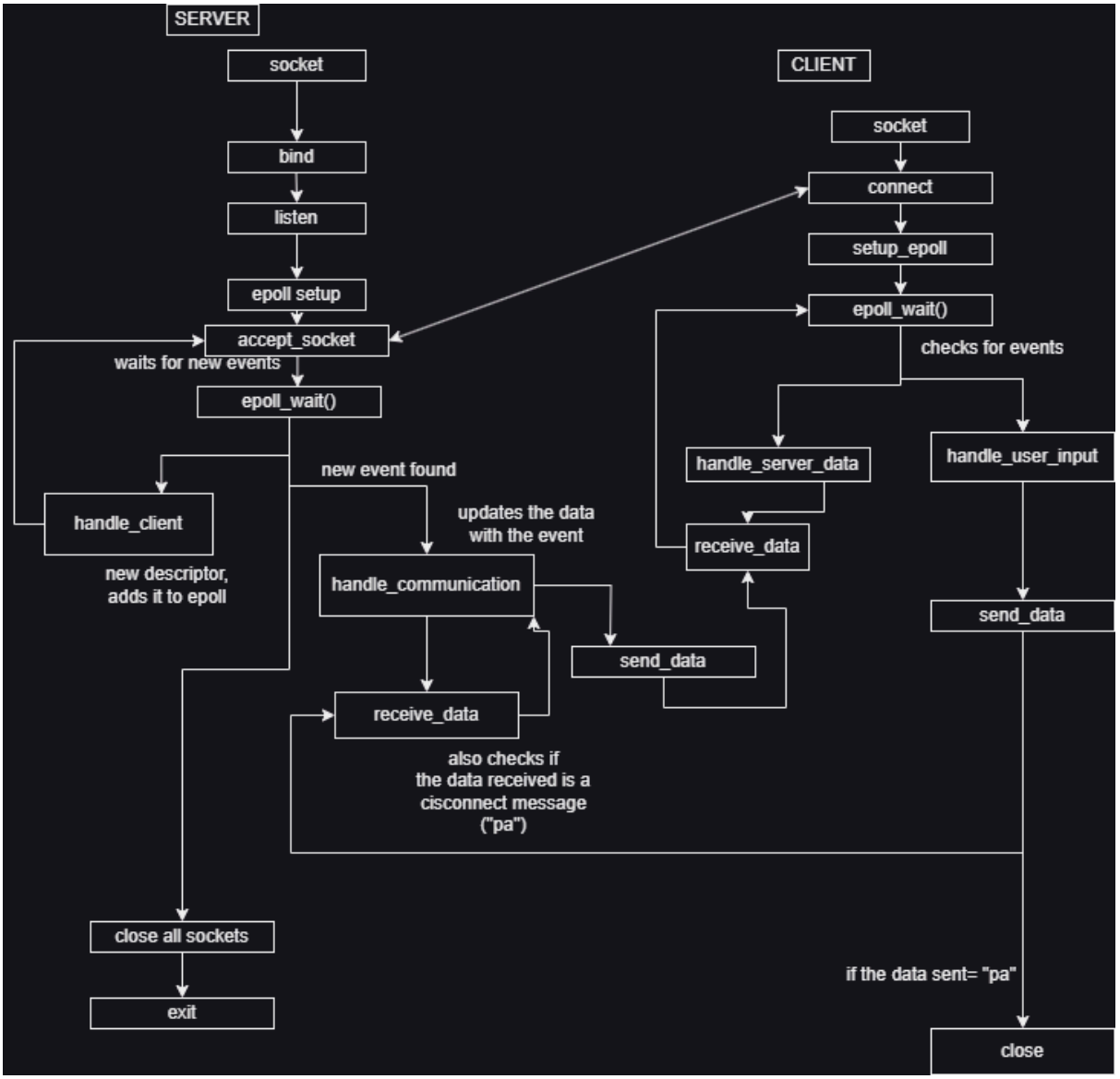
3. Gestionarea Clienților:

- **`client_count`:** Variabilă care ține evidența numărului de clienți conectați.
- **`clients`:** Un vector de pointeri la vectori de booleani. Fiecare vector de booleani reprezintă datele unui client (probabil starea locurilor de parcare asociate acelui client).

4. Manipularea Datelor:

- **`receive_data()`:** Primește date de la client și le stochează într-un șir de caractere.
- **`send_data()`:** Trimite date către client.
- **`stoi()`:** Conversia șirului de caractere primit de la client la un număr întreg, pentru a identifica locul de parcare.

- **Gestionarea Excepțiilor:** Utilizarea blocurilor try-catch pentru a gestiona erorile de conversie a datelor (`invalid_argument`, `out_of_range`).



4. Aspecte de Implementare

Gestionarea Etajelor: Codul folosește un `vector<vector<bool>*> clients` pentru a reprezenta etajele parcării. Fiecare element din vector este un pointer către un `vector<bool>`, care reprezintă starea locurilor de parcare de la un etaj. `true` indică un loc ocupat, iar `false` un loc liber.

```
vector<vector<bool>*> clients;
//...
void handle_client(int epollfd, int serversocket) {
    struct epoll_event ev;
```

```
    int clientsocket=accept(serversocket, NULL, NULL);
    if (clientsocket==-1) {
        cerr<<"accept() failed: "<<strerror(errno)<<endl;
        return;
    }
    vector<bool>* clientData = new vector<bool>;   ///se adaugă câte un vector nou
de fiecare data cand se identifica un nou client
    clients.push_back(clientData);
    ev.events=EPOLLIN;
    ev.data.fd=clientsocket;
    if (epoll_ctl(epollfd, EPOLL_CTL_ADD, clientsocket, &ev)==-1) {
        cerr<<"epoll_ctl() failed"<<strerror(errno)<<endl;
        close(clientsocket);
        delete clientData;
        return;
    }
    cout<<"New client connected (Index: "<<clients.size()-1<<')'<<endl;
//(implementare temporara)
    client_count++;
}
```

Protocol la nivelul aplicației:

- Clientul trimite numărul locului de parcare ca șir de caractere (e.g., "42").
- Serverul convertește șirul la un număr întreg și schimbă starea locului de parcare corespunzător. (urmează sa fie implementată funcția)
- Serverul trimite un mesaj înapoi clientului pentru a confirma acțiunea ("Spot occupied" sau "Spot freed"). (nu este implementată încă funcția)
- Dacă clientul dorește sa inchida conexiunea, clientul trimite mesajul "pa".

```
void handle_communication(int epollfd, int clientsocket) {
    string message;
    receive_data(clientsocket, message);
    if (message=="pa") {
        cout<<"One client requested to end the chat."<<endl;
        close(clientsocket);
        client_count--;
        return;
    }
    bool ok=1;
    try {
        int number = stoi(message);
        cout<<"Locul de parcare "<<number<<" a fost accesat"<<endl; // de modificat
ca sa spuna ca a fost ocupat/eliberat
    }catch (const invalid_argument& e) {
        cerr<<"Invalid value received (not a number)"<<endl;
        ok=0;
    }catch (const out_of_range& e) {
        cerr<<"Number out of range"<<endl;
        ok=0;
    }
    if (ok) {
        message="Parking spot marked successfully!";
    }
    else {
        message="Error: invalid value entered";
```

```
    }
    send_data(clientsocket, message);
}
```

Secțiune de cod inovativă: Utilizarea `epoll` pentru a gestiona mai mulți clienți simultan într-un mod eficient. `epoll` permite serverului să monitorizeze evenimentele de la mai mulți clienți fără a bloca firul de execuție principal.

```
int setup_epoll(int serversocket) {
    int epollfd=epoll_create1(0);
    if (epollfd==-1) {
        cerr<<"epoll_create1() failed "<< strerror(errno)<<endl;
        exit(1);
    }
    struct epoll_event ev;
    ev.events = EPOLLIN;
    ev.data.fd=serversocket;
    if (epoll_ctl(epollfd, EPOLL_CTL_ADD, serversocket, &ev)==-1) {
        cerr<<"epoll_ctl() failed: "<<strerror(errno)<<endl;
        close(epollfd);
        exit(1);
    }
    return epollfd;
}
```

5. Concluzii

O potențială îmbunătățirea ar fi tranziția la IPv6 pentru a avea posibilitatea inscripției datelor transmise și pentru a asigura o capacitate de scalabilitate chiar și mai vastă. Totodata, programul ar putea fi legat de o aplicație mobilă cu un UI simplist și rapid, prin implementarea unei chei de acces API, asigurand compatibilitatea aparentă cu orice platforma.

6. Referințe Bibliografice

Nicholas Day, 20 Jul 2021. *C++ Network Programming*.
https://www.youtube.com/watch?v=gntyAFoZp-E&list=WL

Hoff._world, 12 Mar 2024, All About Epoll - Scalable I/O Syscalls in Linux!
https://www.youtube.com/watch?v=WuwUk7Mk80E&t=57s&pp=ygUFZXBvbGw=

Georgiana Calancea, *Laboratories*
https://profs.info.uaic.ro/georgiana.calancea/laboratories.html