**ACADEMIA TEHNICĂ MILITARĂ “FERDINAND I”**

A yellow eagle with a crown and a shield

Description automatically generated

Proiectarea Sistemelor de Operare Proiect

RPC – Remote Procedure Call

Sd. Sg. Şandru Maria-Laura

Sd. Sg. Gâtan Ema-Maria

Grupa C-113A

Cuprins

[1. Remote Procedure Call 3](#_Toc186921536)

[1.1. Diagrama Mod Sincron 3](#_Toc186921537)

[1.2. Diagrama Mod Asincron 4](#_Toc186921538)

[1.3. Formatul pachetelor 4](#_Toc186921539)

[2. Client 5](#_Toc186921540)

[3. Server 6](#_Toc186921541)

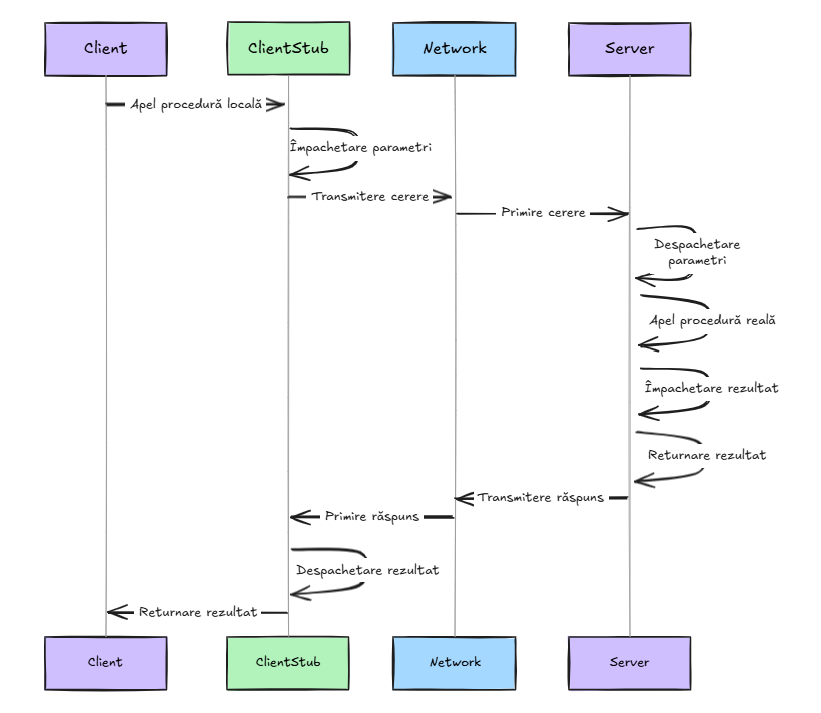
[4. Testare 7](#_Toc186921542)

# Remote Procedure Call

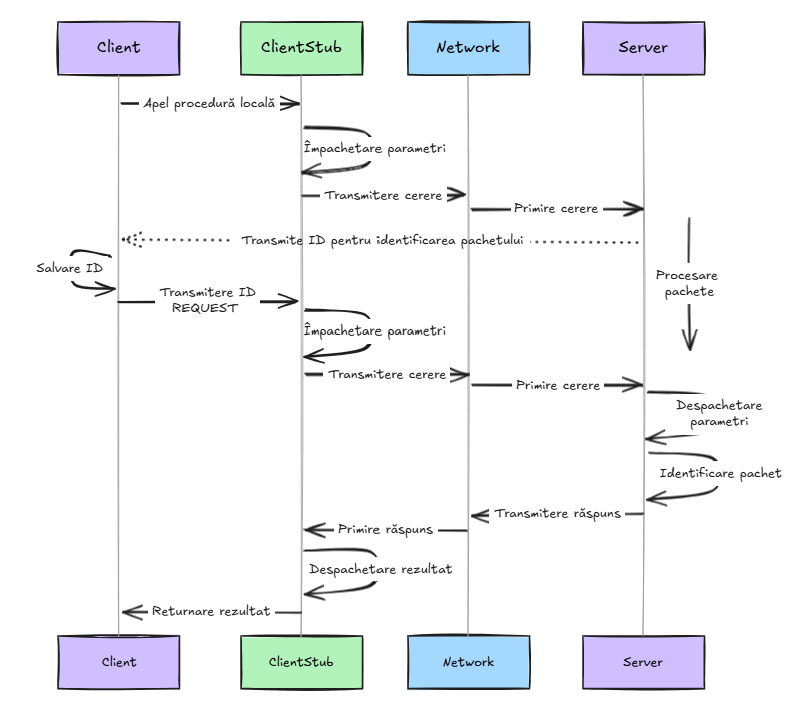
RPC – protocol de comunicație software prin care un program poate să solicite un serviciu de la un alt program aflat pe un alt computer într-o rețea, fără a fi nevoie să cunoască detalii despre rețea. Programul care face solicitarea se numește Client, iar programul care furnizează serviciul se numește Server.  
Acest mecanism este utilizat pentru a apela proceduri de la distanță (funcții) ca și cum ar fi apelate local, simplificând astfel interacțiunea între sistemele distribuite.

Aplicația permite dezvoltatorilor să definească funcții la nivelul serverului, care pot fi ulterior apelate de către client, ca și cum ar fi locale. Aceasta oferă un mecanism eficient de abstractizare, gestionând automat serializarea și deserializarea datelor, transmiterea mesajelor și stabilirea conexiunilor între sisteme folosind socket-uri de tip TCP (fiabilitate). Aplicația oferă suport atât pentru apeluri sincrone, unde clientul așteaptă procesarea informaţiilor transmise înainte de a continua execuția, cât și pentru apeluri asincrone, care permit continuarea execuției fără a aștepta procesarea datelor.

## Diagrama Mod Sincron



## Diagrama Mod Asincron



## Formatul Pachetelor

Structura pachetului include un buffer în care sunt stocate datele transmise, tipul corespunzător pachetului, un offset care indică poziția de extragere a datelor și dimensiunea curentă a pachetului.

Date transmise:

* numere întregi (introducere: AppendInt(int)/extragere: ExtractInt(uint32\_t));
* şiruri de caractere (introducere: AppendString(char)/extragere: ExtractString(char\*));
* float (introducere: AppendFloat(float)/extragere: ExtractFloat(uint32\_t));
* vectori de numere întregi (introducere: AppendIntArray(Packet \*, int \*data, uint32\_t size)/extragere: ExtractIntArray(Packet \*, int \*\*data, uint32\_t \*size));
* vectori de numere reale (introducere: AppendFloatArray(Packet \*, float \*data, uint32\_t size)/extragere: ExtractFloatArray(Packet \*, float \*\*data, uint32\_t \*size)).

Fiecare pachet are asociat un tip, corespunzător scopului creării pachetului:

* SYNC - folosit pentru modul de lucru sincron;
* ASYNC - folosit pentru modul de lucru asincron;
* REQUEST - folosit de către client pentru preluarea datelor procesate în mod asincron;
* ACK - folosit de către server pentru a confirma găsirea si returnarea informaţiilor solicitate de către client.

# Client

La nivelul clientului are loc apelarea funcţiilor prin intermediul unui wrapper care abstractizează comunicarea Client-Server. Wrapper-ul se conectează la procesul care pune la dispoziţie funcţia respectivă, încapsulează numele funcţiei şi parametrii acesteia într-un pachet standardizat, căruia i se atribuie un tip specific modului de lucru sincron sau asincron. Wrapper-ul transmite pachetul pe canalul de comunicație și așteaptă un răspuns, conform modului de lucru asociat.

Modul de lucru sincron:

* wrapper-ul încapsulează numele funcției și parametrii într-un pachet;
* creează conexiunea TCP cu serverul, care pune la dispoziţie funcția respectivă (void callFunction(TCPConnection \*connection, Packet \*sendPacket));
* trimite pachetul pe canalul de comunicație către server (SYNC);
* clientul așteaptă primirea răspunsului de la server (ACK).

Pentru modul de lucru sincron încapsularea denumirii funcţiei şi a parametrilor, dar şi decapsularea datelor se realizează în următoarele funcţii:

* int add(TCPConnection \*connection, int a, int b);
* char \*removeDuplicates(TCPConnection \*connection, char \*buffer, int size);
* int longestAscendingDigitNumber(TCPConnection \*connection, int \*arr, int size);
* int calculateWordFrequency(TCPConnection \*connection, char \*buffer, int bufferSize, char \*word, int wordSize);
* float\* rotateArray(TCPConnection \*connection, float \*arr, int size, int rotations, char \*direction, int directionSize).

Mod de lucru asincron:

* wrapper-ul încapsulează numele funcției și parametrii într-un pachet;
* inițiază conexiunea TCP cu serverul (int callAsyncFunction(TCPConnection \*connection, Packet \*sendPacket));
* pachetul de date este transmis către server (ASYNC);
* Serverul trimite un ID pentru identificarea ulterioară a pachetului (ACK);
* clientul stochează ID-ul;
* răspunsul este gestionat ulterior prin trimiterea unui pachet (REQUEST) ce conține ID-ul pachetului solicitat.

Pentru modul de lucru asincron încapsularea denumirii funcţiei şi a parametrilor, dar şi primirea ID-ului se realizează în următoarele funcţii:

* int addAsync(TCPConnection \*connection, int a, int b);
* int removeDuplicatesAsync(TCPConnection \*connection, char \*buffer, int size);
* int longestAscendingDigitNumberAsync(TCPConnection \*connection, int \*arr, int size);
* int calculateWordFrequencyAsync(TCPConnection \*connection, char \*buffer, int bufferSize, char \*word, int wordSize);
* int rotateArrayAsync(TCPConnection \*connection, float \*arr, int size, int rotations, char \*direction, int directionSize).

Cererile pentru pachetele procesate în modul asincron şi decapsularea datelor se realizează în următoarele funcţii:

* int addRequest(TCPConnection \*connection, int id);
* char \*removeDuplicatesRequest(TCPConnection \*connection, int id);
* int longestAscendingDigitNumberRequest(TCPConnection \*connection, int id);
* int calculateWordFrequencyRequest(TCPConnection \*connection, int id);
* float\* rotateArrayRequest(TCPConnection \*connection, int id).

# Server

La nivelul serverului, se creează un thread dedicat pentru gestionarea cozii de pachete neprocesate (requestsQueue), iar thread-ul principal se ocupă de acceptarea conexiunilor.

Thread-ul principal creează un thread nou pentru fiecare conexiune acceptată, iar acestea procesează pachetele primite de la client în funcție de tipul asociat (SYNC, ASYNC, REQUEST).

Pachete de tip SYNC:

* decapsulează informațiile primite
* identifică funcția solicitată și parametrii acesteia
* apelează funcția locală corespunzătoare
* rezultatul funcției apelate este încapsulat într-un nou pachet (ACK)
* trimite pachetul pe canalul de comunicație către client.

Pachete de tip ASYNC:

* genereaza un ID unic de identificare al pachetului transmis
* incapsuleaza ID-ul intr-un pachet (ACK) si il trimite pe canalul de comunicatie
* salveaza structurat ID-ul si pachetul neprocesat (requestsQueue), de unde va fi preluat si procesat ulteror.

Pachete de tip REQUEST:

* caută ID-ul în coada cu pachete procesate (processedQueue)
* dacă ID-ul exista, se extrage pachetul şi se transmite pe canalul de comunicaţie către client, asocierea pachet, ID va fi ştearsă din coadă
* dacă ID-ul nu exista, are loc o prioritizare în procesarea pachetului. Se caută ID-ul în coada cu pachete neprocesate (requestsQueue), dacă este găsit se va apela funcţia de procesare a pachetului. Dacă pachetul nu este găsit, se presupune că acesta se află în procesare sau a fost procesat şi se încearcă extragerea acestuia din coada cu pachete procesate până este găsit. Ulterior, pachetul va fi transmis către client (ACK).

Thread-ul dedicat pentru administrarea cozii de pachete neprocesate se ocupă de gestionarea unui număr predefinit de thread-uri care pot fi create pentru procesarea pachetelor din coada de pachete neprocesate (requestsQueue).

Acesta monitorizează numărul de thread-uri active și se asigură că nu este depășită limita maximă predefinită de thread-uri care pot fi create pentru procesarea pachetelor. Dacă există pachete în coadă și nu s-a atins numărul maxim de thread-uri active, se atribuie câte un pachet și ID-ul asociat unui thread nou creat pentru a-l procesa. După procesare, asocierile sunt salvate în coada de pachete procesate.

Dacă există pachete în așteptare și thread-uri disponibile, thread-ul dedicat atribuie pachetele acestora, asigurând astfel procesarea eficientă a cererilor.

Serverul pune la dispoziţie implementarea următoarelor funcţii:

* void removeDuplicates(char \*buffer, int \*size) - ştergerea duplicatelor alăturate dintr-un şir
* int longestAscendingDigitNumber(int \*arr, int size) - determinarea numărului cu cea mai lungă secvenţă de numere crescătoare dintr-un vector de numere întregi
* int calculateWordFrequency(char \*buffer, char \*word) - frecvenţa unui cuvânt într-un şir
* void rotateArray(float \*arr, int size, int k, char \*direction) - rotirea unui vector de numere reale cu k poziţii în direcţia left /right
* int add(int a, int b) - suma a două numere întregi.

# Testare

Pentru testarea conectivităţii clienţilor şi apelului de proceduri în mod sincron, cât şi asincron, se rulează scriptul run.sh.