**ACADEMIA TEHNICĂ MILITARĂ “FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ŞI SECURITATE CIBERNETICĂ**

**Specializarea: Calculatoare şi sisteme informatice pentru apărare şi securitate naţională**A yellow eagle with a crown and a shield

Description automatically generated

Proiectarea Sistemelor de Operare: Proiect

Processing System

2024

Nume și prenume: Heașcă Antonio-Ciprian

Sîrbu Bianca-Georgiana

Grupa: C-113C

Cuprins

[I. Introducere 3](#_Toc182091153)

[A. Scopul proiectului 3](#_Toc182091154)

[B. Lista definițiilor 3](#_Toc182091155)

[II. Descrierea generală a proiectului 4](#_Toc182091156)

[III. Contex si utilitate practică 5](#_Toc182091157)

[IV. Funcționalitățile principale ale aplicației 6](#_Toc182091158)

[V. Dependințe și Limitări 6](#_Toc182091159)

# Introducere

## Scopul proiectului

Scopul acestui proiect este de a dezvolta un sistem software de tip client-server pentru execuția sarcinilor între agenți multipli, gestionate de un server centralizat. Sistemul va permite agenților să se înregistreze la server, iar serverul va administra request-urile primite de agenți=clienți, redirecționandu-le către agenți disponibili.

## Lista definițiilor

Pentru o mai bună înțelegere a documentului, se vor defini termenii importanți utilizați în sistem:

**Server:** componenta centrală care primește sarcini de la clienți, distribuie si gestionează sarcinile astfel încât să le atribuie agenților disponibili. În cazul în care un agent nu reușește sa execute acea sarcină, serverul se ocupă de redistribuirea sarcinii.

**Agent:** program software instalat pe mai multe stații de lucru care se înregistrează pe server, primește sarcini și le execută.

**Socket**: Metodă de comunicare utilizată între server și agenți, bazată pe protocoalele de rețea, de obicei TCP/IP, pentru a permite transferul de date bidirecțional. Socketurile sunt folosite pentru înregistrarea agenților/clienților, trimiterea de sarcini și raportarea stării execuției.

Protocol de Comunicare: Protocolul specific utilizat pentru schimbul de mesaje între server și agenți, care poate include structura mesajelor, porturi și metode de criptare a datelor. În acest sistem, protocoalele sunt esențiale pentru autentificarea agenților și securizarea transferurilor de date.

# Descrierea generală a proiectului

Sistemul propus este un sistem distribuit de execuție a sarcinilor care include un server centralizat și agenți distribuiți instalați pe diverse stații de lucru.

Sistemul funcționează astfel:

* Server
* Cererile de execuție a unor sarcini vor fi trimise de clienți prin intermediul unui protocol stabilit.
* În momentul în care serverul primește sarcina, acesta o procesează și verifică disponibilitatea agenților.
* Serverul va aloca sarcinile în funcție de agenții disponibili și va gestiona starea acestora(disponibil/indisponibil).
* Comunicarea dintre agenți si server se va realiza prin socket-uri iar implementarea va fi în limbajul de programare C.
* Pe parcursul execuției sarcinilor de către agent, va fi menținută o conexiune activă cu serverul.
* Agent
  + Fiecare agent este un program instalat pe o stație de lucru care se conectează la server, se înregistrează(primește un identificator) și așteaptă alocarea de sarcini.
  + Din momentul în care agentul se conectează la server, acesta intră în starea de așteptare.
  + După ce primește o sarcină, agentul o **execută local** și raportează serverului rezultatul sau eventualele erori întâlnite.
  + Cât timp agentul execută o sarcină acesta este indisponibil și nu va primi o nouă sarcină până când nu va termina executiția celei curente.
  + Agenții se pot deconecta și reconecta fără să afecteze serverul, iar serverul va menține o evidență a agenților activi.

**Context:**

Agentul va fi reprezentat de către un executabil care poate rula pe mai multe stații disponibile. Agentul poate trimite request-uri către server pentru a executa o anumită sarcină, dar poate să și primească task-uri de la server, nu în același timp.

Exemplu: Agentul A trimite un request către server cu o sarcină de a rula un script .Serverul va primi request-ul,îl va procesa și îl va transmite mai departe către un agent disponibil. Agentul B va rula scriptul, iar rezultatul va fi transmis către server care la rândul acestuia va trimite răspunsul către agentul A (serverul -thread-ul- va reține informații despre agentul care a trimis cererea și cel care a rezolvat-o).

# Context si utilitate practică

Acest proiect ar putea fi util în cadrul unei mari companii care se ocupă cu analize complexe de date, procesare imagini sau rulare executabile ce necesită resurse hardware ridicate. Într-un astfel de mediu, task-urile de procesare ar putea fi distribuite între mai multe stații de lucru pentru a reduce încărcarea pe un singur server central și a accelera timpii de execuție.

**Scenariu:**

O organizație financiară analizează constant tranzacții iar precizia lor trebuie sa fie foarte exactă, totul executându-se aproape în timp real pentru a detecta anumite activități fraudoase.

Astfel, ne putem imagina urmatorul flux:

**Serverul** primește constant detalii despre tranzacțiile care au loc pe anumite

stații(agenți) care le apațin și imediat șe verifică integritatea datelor.

Agenții, instalați pe mai multe stații de lucru din cadrul companiei, procesează aceste date, utilizând algoritmi de detectare a anomaliilor, dar acești algoritmi necesită putere de procesare ridicată și distribuie saricinile astfel încât să eficientizeze întreg procesul.

Avantaje:

* **Reducerea Timpului de Procesare**: deoarece sarcinile sunt distribuite către agenți capabili sa le execute, întreg procesul este eficientizat
* **Scalabilitate și Flexibilitate: Sistemul poate gestiona un număr mare de agenți care sunt specializați pe executia acestor sarvcini**
* **Eficiență în Utilizarea Resurselor**: Agenții folosesc resursele locale ale stațiilor de lucru și nu încarcă serverul central cu task-uri individuale
* **Fiabilitate și Continuitate**: Dacă unul dintre agenți devine indisponibil, serverul poate redistribui task-urile, fara a compromite întregul server.

# Funcționalitățile principale ale aplicației

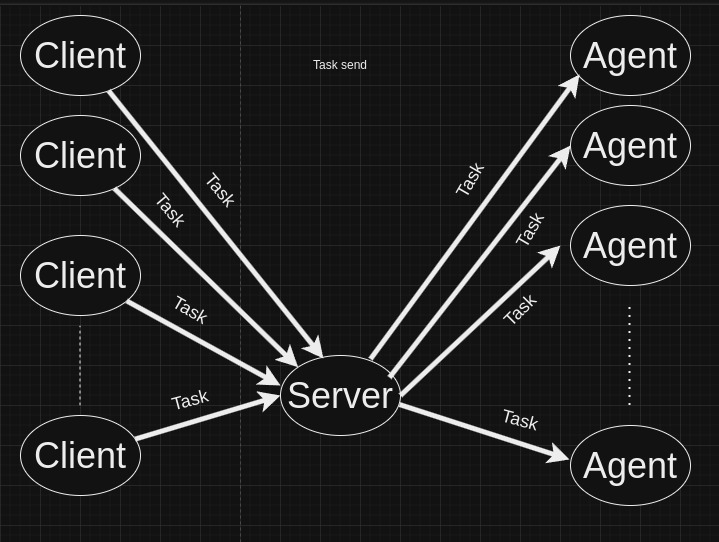
* Modul înregistrare agenți prin socket-uri folosind thread-uri (gestionare multiplă a agenților)
* Modul send/receive: sistemul poate gestiona un număr mare de agenți
* Modul verificare conectivitate ( fiecărui agent conectat i se va impune o limită de timp în care dacă nu trimite un “ping” către server , serverul îl va “dropa”)

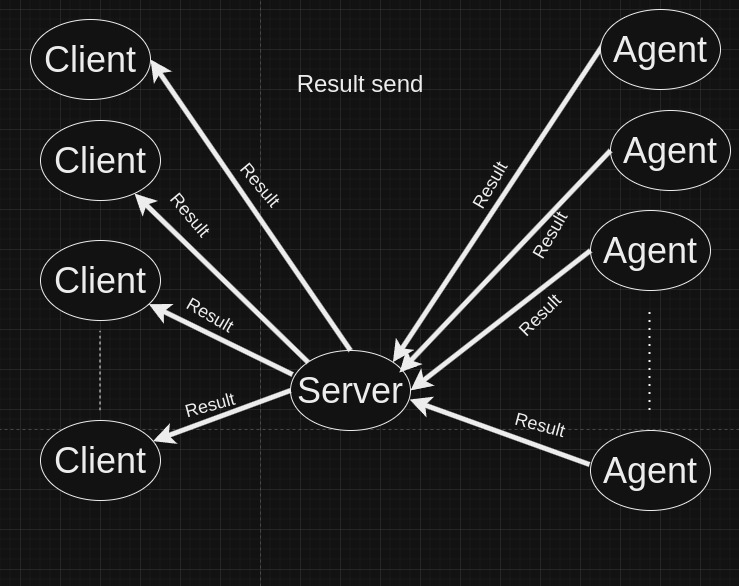
# Dependințe și Limitări

**Cerințe de Sistem**:

* **Platforme Hardware**: Sistemul necesită stații de lucru sau servere cu putere de procesare adecvată pentru a găzdui agenții și serverul centralizat. De preferat, hardware-ul să dispună de o memorie suficientă și capacitate de procesare pentru a suporta sarcinile primite.
* **Platforme Software**: Sistemul este proiectat să ruleze pe sisteme de operare compatibile (ex: Linux, Windows). Serverul și agenții sunt dezvoltați folosind limbajul de programare C și biblioteci compatibile cu socket-uri de rețea.
* **Rețea Stabilă**: Sistemul are nevoie de o conexiune de rețea stabilă între server și agenți pentru comunicarea prin socketuri. Deconectările agenților pot afecta temporar performanța, însă reconectarea automată este gestionată de server.

# Diagramă mecanism Processing Systems





# Flux de bază

Clienții trimit task-uri către server, care la rându-i asignează agenților disponibili care îndeplinesc condițiile, task-urile.

Serverul va conține o coadă de task-uri primite de la clienți, precum și o coadă de agenți. Task-ul, clientul și agentul vor fi modelați cu ajutorul structurilor. Structura Client va fi inclusă în structura Task.

Printre câmpurile structurilor Task și Agent se vor afla câmpuri care vor categorisi task-urile în funcție de “resursele necesare”, iar agenții în funcție de tipul de task-uri pe care le pot executa eficient.