MULTITHREADED PROXY-SERVER

PROIECT - PSO

Contribuitori:

Std.sg. Enache Stefan

Grupa C113-D

CUPRINS:

1. Introducere

1.1. Scopul proiectului

Proiectul implica dezvoltarea unui program de tip Proxy-Server a carui sarcina este sa trimita mai departe, in numele clientului si catre serverul de origine al acestuia, request-uri de tip:

- GET
- POST
- CONNECT

Server-ul trebuie sa poata gestiona mai multe solicitari simultane, de aceea se vor utiliza mai multe thread-uri pentru a trata in paralel request-urile de pe fiecare conexiune.

Motivul principal pentru implementare consta in permiterea accesului la surse externe din interiorul retelei protejate de firewall sau direct inaccesibile si deserveste drept gateway pentru intermedierea traficului de retea al protocolului aplicatiei.

2. Descrierea generală a produsului software

2.1. Descrirea produsului software

Server-ul proxy va fi utilizat pentru securitatea comunicarii in internet si eficientizarea timpului de raspuns prin caching, principiul de baza al functionarii fiind de a primi cereri de client, aceste cereri sunt analizate si se executa trimiterea catre serverele tinta. Acesta actioneaza asupra nivelului 7 din modelul OSI (aplicatie) pentru a analiza cererile primite. Pe langa acest proxy de aplicatie, exista si alte independente de aplicatie, care furnizeaza numai pachete de transport fara cunostintele protocoalelor de nivel de aplicatie. Folosirea lor necesita totusi utilizarea unui protocol de comunicare specific pentru a comunica cu server-ul proxy.

2.2. Detalierea platformei HW/SW

Produsul este alcatuit dintr o singura aplicatie software, ce va fi atajata retelei in realizarea legaturii in acest spatiu dintre sarcina de lucru intre furnizorii unei resurse sau serviciu si solicitantii de servicii. Ca instrument de lucru vom folosi : VS Code , intr un mediu Linux.

2.3. Constrângeri

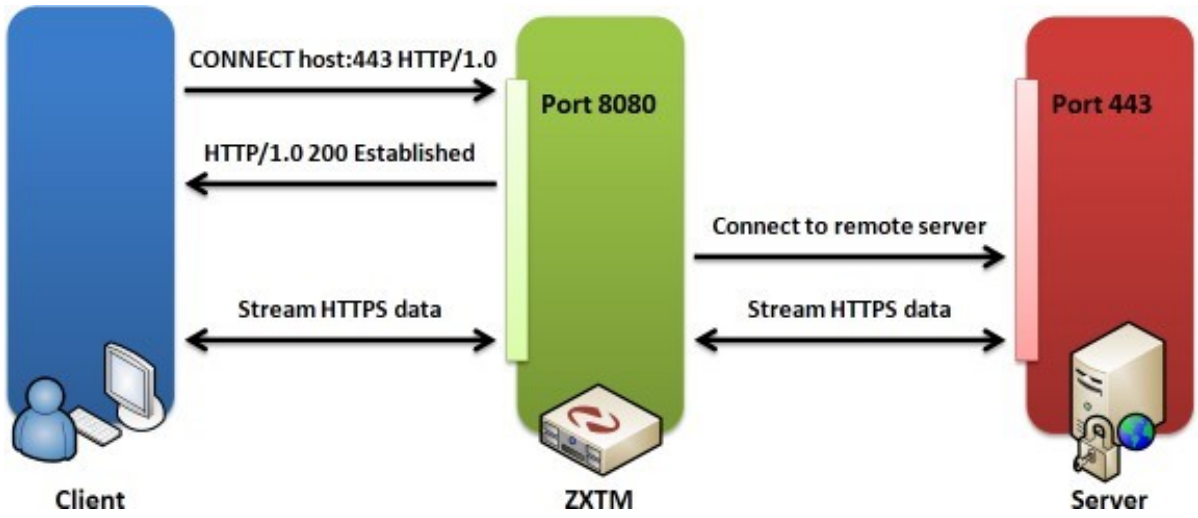
- Serverul implementeaza suport complet pentru standart-ul HTTP 1.0, adica metodele GET , POST, HEAD si accepta un subset al standartului HTTP 1.1 pentru a permite comunicarea fara probleme a implementarilor existente ale clientului si serverului(de exemplu: browsere web si servere web).

- Serverul este eficient in ceea ce priveste consumul de timp al procesorului. Aceasta eficacitate este redata de proxy pentru ca realizeaza toate operatiie de intrare si iesire din retea. Aplicatia petrecand de cele mai multe ori asteptand sosirea de date suplimentare sau conexiuni ulterioare.

3. Detalierea cerințelor software

3.1. Cerințele funcționale

- Permiterea a cati mai multi clienti sa poata accesa serverul proxy.
- Proxy pentru conexiuni cu site-uri care folosesc protocolul https: Se transmite request-ul **CONNECT** pentru a stabili o conexiune securizata, apoi se face doar forward intre browser si server fara a cripta/decripta traficul care se transmite.



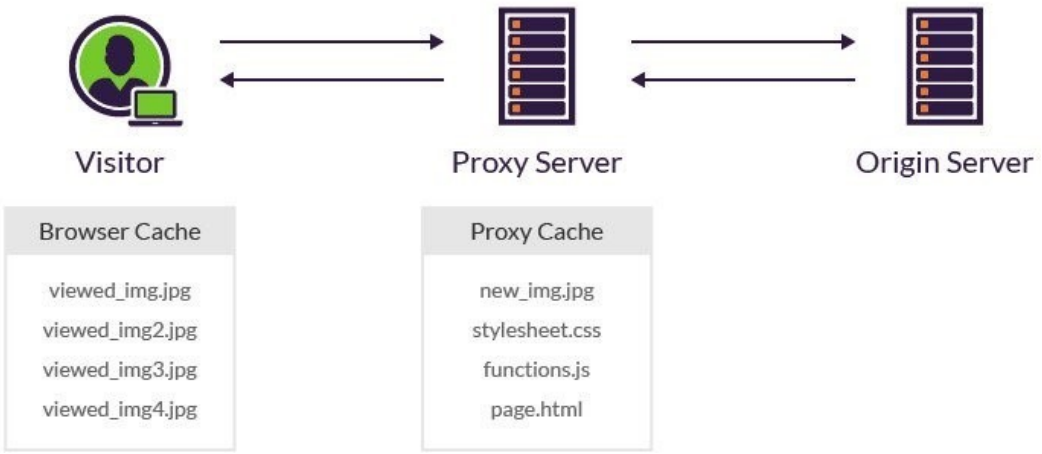
- Pentru implementarea functiilor de comunicare a cererilor/respunsurilor in retea, intre Client-proxy-server web a fost definit **namespace-ul Utils**.
- Extragerea informatiilor din header-ele unui raspuns se realizeaza prin functiile definite in **ResponseParser**
- Toate request-urile si raspunsurile pe care proxy-ul le proceseaza vor fi salvate in fisierul de **proxy.log**.Fiecare linie fisier respecta urmatorul format:
  - ✓ ID: pentru fiecare conexiune va fi asignat un id, prin acest id putem identifica corespondenta dintre request si raspuns in fisierul de log
  - ✓ STATUS LINE
  - ✓ HOST
  - ✓ TIME
- Scrierea in fisierul de log de catre fiecare thread va fi controlata printr-un semafor *logMutex* pentru a asigura consistenta fisierului

- Proxy-ul poate bloca accesul la site-uri, site-urile blocate vor fi setate prin scriere in fisierul block.config a unei liste de host-uri. In cazul in care se incarca accesul la o pagina blocata se va trimite catre browser o pagina specifica de eroare (403 Forbidden).

- Arhitectura programata pentru mai multe fire de executie.Pentru a spori performantele aplicatiei am implementat un Thread pool.Se reduce overhead-ul de a crea si distruge thread-urile prin crearea initiala a unor thread-uri care sa ruleze pe tot parcusul executiei.Acest lucru ne permite si sa controlam numarul maxim de conexiuni procesate intr-un singur moment de catre server.

- Tot pentru sporirea performantelor si eficientizarea timpului de raspuns, server-ul stocheaza raspunsurile la care se poate cache.Pentru a gestiona cache-ul am implementat clasa **WebCache.**Prin aceasta am implementat un mecansim de tip **LRU Cache** cu scopul de a eficientiza consumul de memorie prin stocarea doar cererilor care au fost utilizate cel mai des.

- In clasa Reponse si in Cache salvam ***politicile de cache***.Header-ul ***Cache-Control*** este utilizat pentru a spceifica politicile de cache intr-un raspuns HTTP.Acesti parametri ne permit sa vedem daca putem stoca raspunsul in **cache-ul Proxy** sau pentru cat timp.



- Dupa pornirea functiei main() se deschide socket-ul acceptand noi conexiuni.Noua comunicare de intrare este verificata daca nu depaseste limita maxima de conexiuni.

3.2. Cerințele ne-funcționale

- Produsul poate functiona pe sisteme de operare distribuite Linux.
- Cerintele de performanta pentru functionarea optima sunt minime.
- Aplicatia va avea nevoie de o comunicatie intr o retea pentru realizarea legaturilor intre utilizatorii acesteia.