Simple Web Server

Perju Rares-Andrei

Facultatea de Informatica Iasi

1.Introducere

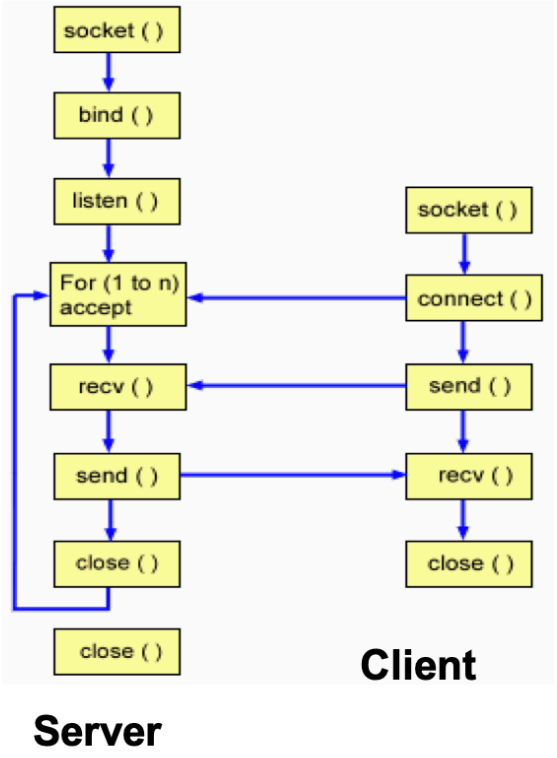
Proiectul Simple Web Server propune o aplicatie de comunicare intre un server local si un browser oarecare . Rolul acestei aplicatii este de a vizualiza , dintr-o pagina web , fisiere de tip .html sau .txt .Putem observa aici că ideea de web server presupune și noțiunea de hosting (găzduire), asta deoarece serverul trebuie să dețină datele pe care urmează să le returneze la cerere.

Relația este următoarea: utilizatorul (clientul) aflat în dreptul unui computer pe care are instalată o aplicație tip browser solicită (serverului) prin intermediul unui url o anumită pagină web; serverul rulează anumite linii de cod și returnează un rezultat. Descrierea de mai sus se potrivește perfect în cazul site-urilor (paginilor web) statice (adică cele bazate în exclusivitate doar pe limbajele html și css).Există însă așa numitele site-uri dinamice care au în compunere alături de limbajul rudimentar de afișare a paginii web și un limbaj de comunicare între serverul web și o bază de date. De această dată relația este următoarea: utilizatorul (clientul) aflat în dreptul unui computer pe care are instalată o aplicație tip browser solicită (serverului) prin intermediul unui url o anumita pagină web; serverul web verifică solicitarea și prin intermediul unui limbaj de programare special interoghează o bază de date, dacă anumite condiții sunt întrunite, baza de date returnează serverului web datele solicitate care la rândul lui furnizează datele mai departe clientului inițial. În această ultimă relație între web server și utilizator sunt transmise doar informații destinate afișării în browser într-o formă prietenoasă a informațiilor solicitate. Până să ajungă la utilizator, așa cum am văzut mai sus, serverul web prin intermediul unui scripting special schimbă o serie de informații cu o bază de date stocată.

2.Tehnologii utilizate

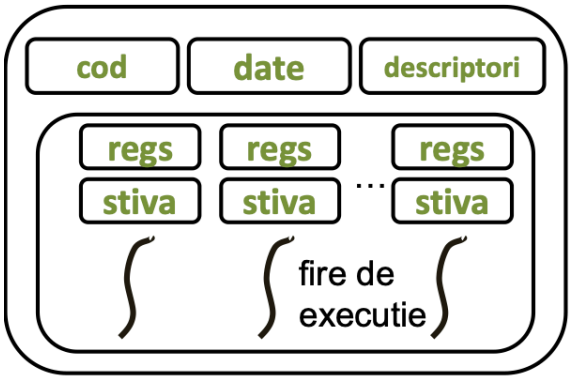
2.1.1 TCP/IP

Tipul de protocol necesar realizarii comunicarii este TCP (Transmis- sion Control Protocol) concurent , deoarece realizeaza o comunicare sig- ura(datele sunt trimise in pachete acestea fiind transmise independent si sunt reasamblate odata ce ajung la destinatie pentru a ajunge la forma originala) , ordonata(toate pachetele sunt reasamblate in oridine) si farapierdere de date . Acest protocol permite accesarea simultana de pe maimulte browsere .



Model server/client TCP

De asemenea serverul se bazeaza pe implementarea sockets multithreading ce ofera conectarea simultana si concurenta a mai multor utilizatori .



Procese cu mai multe fire de executie

2.1.2 Nivelui

Diversele cerințe au condus la alegerea a patru niveluri pentru modelul TCP/IP: Aplicație, Transport, Rețea (sau Internet) și Acces la Rețea.

I. Nivelul Aplicație

Nivelul aplicație se referă la protocoalele de nivel înalt folosite de majoritatea aplicațiilor, precum terminalul virtual (TELNET), transfer de fișiere (FTP) și poștă electronică (SMTP). Alte protocoale de nivel aplicație sunt DNS (sistem de nume de domeniu), NNTP sau HTTP.

II.Nivelul Transport

Este identic cu cel din modelul OSI, ocupându-se cu probleme legate de siguranță, control al fluxului și corecție de erori. El este proiectat astfel încât să permită comunicarea între entitățile pereche: sursă, respectiv, destinație. În acest sens au fost definite două protocoale capăt-la-capăt.

Primul din ele, **TCP (Transmission Control Protocol)**. El este un protocol sigur orientat pe conexiune care permite ca un flux de octeți trimiși de pe o mașină să ajungă fără erori pe orice altă mașină din inter-rețea. Acest protocol fragmentează fluxul de octeți în mesaje discrete și pasează fiecare mesaj nivelului internet. TCP tratează totodată controlul fluxului pentru a se asigura că un emițător rapid nu inundă un receptor lent cu mai multe mesaje decât poate acesta să prelucreze.

Al doilea protocol din acest nivel, **UDP (User Datagram Protocol)**, este un protocol nesigur, fără conexiuni, destinat aplicațiilor care doresc să utilizeze propria lor secvențiere și control al fluxului. Protocolul UDP este de asemenea mult folosit pentru interogări rapide întrebare-răspuns, client-server și pentru aplicații în care comunicarea promptă este mai importantă decât comunicarea cu acuratețe, așa cum sunt aplicațiile de transmisie a vorbirii și a imaginilor video.

III.Nivelul Retea (Internet)

Scopul inițial al nivelului rețea ("Internet Protocol") era să asigure rutarea pachetelor în interiorul unei singure rețele. Odată cu apariția interconexiunii între rețele, acestui nivel i-au fost adăugate funcționalități de comunicare între o rețea sursă și o rețea destinație.

În stiva TCP/IP, protocolul IP asigură rutarea pachetelor de la o adresă sursă la o adresă destinație, folosind și unele protocoale adiționale, precum ICMP sau IGMP. Determinarea drumului optim între cele două rețele se face la acest nivel.

Comunicarea la nivelul IP este nesigură, sarcina de corecție a erorilor fiind plasată la nivelurile superioare (de exemplu prin protocolul TCP). În IPv4 (nu și IPv6), integritatea pachetelor este asigurată de sume de control.

IV.Nivelul Acces la retea

Se ocupă cu toate problemele legate de transmiterea efectivă a unui pachet IP pe o legătură fizică, incluzând și aspectele legate de tehnologii și de medii de transmisie, adică nivelurile OSI Legătură de date și Fizic.

2.2. HTTP

Acesta este un alt protocol folosit pentru realizarea proiectului. Hypertext Transfer Protocol (HTTP) este metoda cea mai des utilizată pentru accesarea informațiilor în Internet care sunt păstrate pe servere World Wide Web (WWW).

Protocolul HTTP este un protocol de tip text, fiind protocolul "implicit" al WWW. Adică, dacă un URL nu conține partea de protocol, aceasta se consideră ca fiind http.

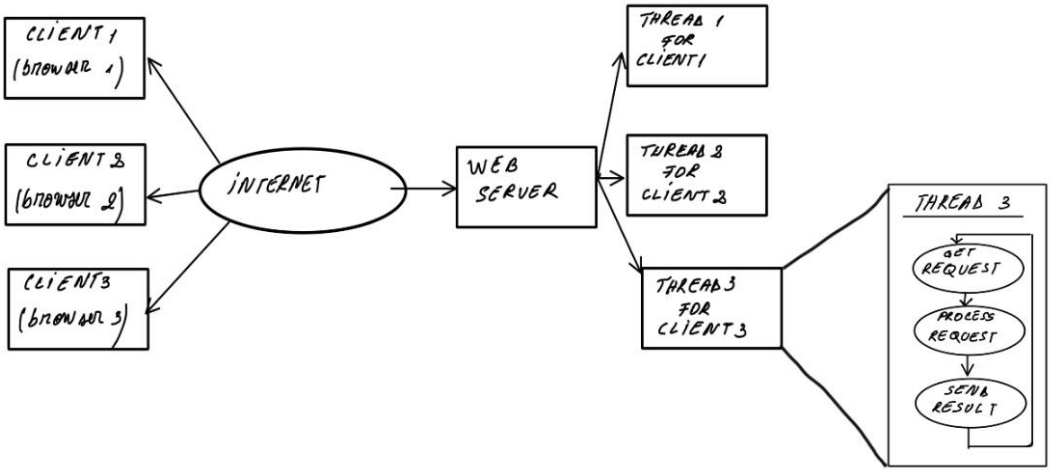
HTTP presupune că pe calculatorul destinație rulează un program care înțelege protocolul. Fișierul trimis la destinație poate fi un document HTML (abreviație de la HyperText Markup Language), un fișier grafic, de sunet, animație sau video, de asemenea un program executabil pe server-ul respectiv sau și un editor de text.

După clasificarea după modelul de referință OSI, protocolul HTTP este un protocol de nivel aplicație.

Realizarea și evoluția sa este coordonată de către World Wide Web Consortium (W3C).

HTTP oferă o tehnică de comunicare prin care paginile web se pot transmite de la un computer aflat la distanță spre propriul computer. Dacă se apelează un link sau o adresă de web cum ar fi http://www.example.com, atunci se cere calculatorului host să afișeze o pagină web (index.html sau altele). În prima fază numele (adresa) www.example.com este convertit de protocolul DNS într-o adresă IP. Urmează transferul prin protocolul TCP pe portul standard 80 al serverului HTTP, ca răspuns la cererea HTTP-GET. Informații suplimentare ca de ex. indicații pentru browser, limba dorită ș.a. se pot adăuga în header-ul (antetul) pachetului HTTP. În urma cererii HTTP-GET urmează din partea serverului răspunsul cu datele cerute, ca de ex.: pagini în (X)HTML, cu fișiere atașate ca imagini, fișiere de stil (CSS), scripturi (Javascript), dar pot fi și pagini generate dinamic (SSI, JSP, PHP și ASP.NET). Dacă dintr-un anumit motiv informațiile nu pot fi transmise, atunci serverul trimite înapoi un mesaj de eroare. Modul exact de desfășurare a acestei acțiuni (cerere și răspuns) este stabilit în specificațiile HTTP.

3.Diagrama aplicatiei



4.Secvente relevante de cod

```
17
18    #define SERVER_PORT 8080     // Portul standard HTTP pentru serverul web
19
```

```
38
39    /* crearea unui socket */
40    if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
41      {
42        perror ("[server]Eroare la socket().\n");
43        return errno;
44      }
```

```
bzero(&servaddr,sizeof(servaddr));              // Initializarea structuri de date a serverului
servaddr.sin_family      =AF_INET;              // Stabilirea familiei de socket-uri
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);     // Acceptarea oricarei adrese
servaddr.sin_port        =htons(SERVER_PORT);   // Uilizam un port utilizator
```

```
61      while(1)        // Servim in mod concurent clientii folosind thread-uri
62      {
63          //int i=1;
64          struct sockaddr_in addr;
65          socklen_t addr_len;
66          char client_addr[MAXLINE+1];
67
68
69          printf("\nAsteptam conexiunea la portul %d\n\n", SERVER_PORT);
70          fflush(stdout);
71          client=accept(listenfd, (SA *) &addr, &addr_len);       // Asteptam un client intr-o stare blocanta
72
73          inet_ntop(AF_INET,&addr,client_addr,MAXLINE);
74          printf("Client connection: %s\n",client_addr);
75
76          memset(recvline,0,MAXLINE);
77
78          pthread_t t;
79          int *pclient= malloc(sizeof(client+1));
80          *pclient = client;
81          pthread_create(&t , NULL , raspuns ,pclient);
82
```

```
120         // pregatirea fisierului text pentru a fi afisat in browser
121
122         fp = fopen ( "index.txt" , "rb" );
123         if( !fp ) perror("index.txt"),exit(1);
124
125         fseek( fp , 0L , SEEK_END);
126         lSize = ftell( fp );
127         rewind( fp );
128
129         /* aloc memorie pentru continut*/
130         buffer = calloc( 1, lSize+1 );
131         if( !buffer ) fclose(fp),fputs("alocarea memoriei a esuat",stderr),exit(1);
132
133         /* copiez fisierul in buffer*/
134         if( 1!=fread( buffer , lSize, 1 , fp) )
135         fclose(fp),free(buffer),fputs("toata citirea a esuat",stderr),exit(1);
136
137         fclose(fp);
138
139
140         //char message[] = "HTTP/1.1 200 Okay\r\nContent-Type: text/html; charset=ISO-8859-4 \r\n\r\n<h1>He
141         int length = strlen(buffer); // Plus 1 for  null terminator
142         int send_res = send(client, buffer, length, 0); // Flag = 0
143
144         free(buffer);
145
146         //snprintf((char*)buff,sizeof(buff),"http/1.0 200 OK\r\n\r\n",s);
147
148         write(client,(char*)buff,strlen((char *)buff));
149
150         close(client);
151         return NULL;
152 }
```

5.Scenariu de utilizare

Cel mai bun scenariu de utilizare este gazduirea uniu site web , ce permite publicarea unui site, realizat in format HTML , pe internet permitant altor persoane sa il aceseze si sa il vizualizeze in mod simultan .

6. Inbunatatiri ale aplicatiei

O modalitate de a inbunatati functionalitatea si siguranta datelor transmise este prin folosirea protocolului HTTPS (Secure Hyper Text Transfer Protocol) , care reprezintă protocolul HTTP încapsulat într-un flux SSL/TLS care criptează datele transmise de la un browser web la un server web, cu scopul de a se oferi o identificare criptată și sigură la server. Conexiunile HTTPS sunt folosite în mare parte pentru efectuarea de operațiuni de plată pe World Wide Web și pentru operațiunile "sensibile" din sistemele de informații corporative. HTTPS nu trebuie confundat cu Secure HTTP (S-HTTP) specificat în RFC 2660. HTTPS este utilizat pentru 19,24% din totalul domeniului românesc.

HTTPS este un protocol de comunicație destinat transferului de informație criptată prin intermediul WWW. A fost dezvoltat din necesitatea de a proteja de intruși transferul datelor prin HTTP - un protocol "clear-text", prin care datele de pe server-ul web sunt transmise browser-ului client în clar, posibilitățile de a intercepta acest transfer constituind tot atâtea posibilități de a accesa și utiliza fără restricții informațiile respective. HTTPS nu este altceva decât HTTP "încapsulat" cu ajutorul unui flux SSL/TLS - datele sunt criptate la server înainte de a fi trimise clientului, astfel încât simpla interceptare a acestora pe traseu să nu mai fie suficientă pentru a avea acces la informații. HTTPS este în același timp o metodă de autentificare a server-ului web care îl folosește, prin intermediul așa-numitelor "certificate digitale" - o colecție de date pe care un browser o solicită server-ului pentru a putea începe transferul criptat; dacă certificatul este emis de o autoritate cunoscută (de exemplu VeriSign), browser-ul poate fi sigur că server-ul cu care comunică este ceea ce pretinde a fi.

6.Bibliografie

1. MatrialeCurs : https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
2. Materiale seminar: https://profs.info.uaic.ro/~ioana.bogdan/
3. Wikipedia : https://ro.wikipedia.org/wiki/Transmission_Control_Protocol
4. https://ro.wikipedia.org/wiki/Hypertext_Transfer_Protocol