

Programare de sistem și de rețea

Lucrare de laborator nr.7

Proiectarea unui server web simplu

Scopul lucrării

Se dorește implementarea unui server web simplificat care funcționează doar cu pagini web statice.

Partea 1 : Protocolul HTTP

Introducere

Un server și un client web utilizează un protocol special care se numește HTTP (Hypertext Transfer Protocol) și care este un protocol de comunicație responsabil cu transferul de hipertext (text structurat ce conține legături) dintre un client (de regulă, un navigator) și un server web, interacțiunea dintre acestea (prin intermediul unei conexiuni TCP persistente pe portul 80) fiind reglementată de [RFC 2616](#). HTTP este un protocol fără stare, pentru persistența informațiilor între accesări fiind necesar să se utilizeze soluții adiacente (cookie, sesiuni, rescrierea -urilor, câmpuri ascunse).

Principalele concepte cu care lucrează acest protocol sunt *cererea* și *răspunsul*.

- **cererea** este transmisă de client către serverul web și reprezintă o solicitare pentru obținerea unor resurse (identificate printr-un); aceasta conține denumirea metodei care va fi utilizată pentru transferul de informații, locația de unde se găsește resursa respectivă și versiunea de protocol;
- **răspunsul** este transmis de serverul web către client, ca rezultat al solicitării primite, incluzând și o linie de stare (ce conține un cod care indică dacă starea comenzii) precum și alte informații suplimentare

Structura unei cereri HTTP

O cerere HTTP conține una sau mai multe linii de text , precedate în mod necesar de denumirea metodei specificând operația ce se va realiza asupra conținutului respectiv:

DENUMIRE METODĂ	DESCRIERE
GET	descărcarea resursei specificate de pe serverul web pe client; majoritatea cererilor către un server web sunt de acest tip GET /page.html HTTP/1.1 Host: www.server.com
HEAD	obținerea antetului unei pagini Internet, pentru a se verifica parametrii acesteia

	sau doar pentru a testa corectitudinea unui
PUT	încărcarea resursei specificate de pe client pe serverul web (cu suprascrierea acesteia, în cazul în care există deja); trebuie specificate și datele de autentificare, utilizatorul respectiv trebuind să aibă permisiunile necesare pentru o astfel de operație
POST	transferul de informații de către client cu privire la resursa specificată, acestea urmând a fi prelucrate de serverul web POST /page.html HTTP/1.1 Host: www.server.com attribute1=value1&...&attributen=valuen
DELETE	ștergerea resursei specificate de pe serverul web, rezultatul operației depinzând de permisiunile pe care le deține utilizatorul ale cărui date de autentificare au fost transmise în antete
TRACE	solicitare de retransmitere a cererii primite de serverul web de la client, pentru a se testa corectitudinea acesteia
CONNECT	rezervat pentru o utilizare ulterioară
OPTIONS	interogare cu privire la atributele serverului web sau ale unei resurse găzduite de acesta

Capitalizarea este importantă atunci când se precizează denumirea metodei folosite, făcându-se distincție între minuscule și majuscule.

Cel mai frecvent, se utilizează metodele **GET** (folosită implicit, dacă nu se specifică altfel) și **POST**.

GET vs. POST

Deși atât metoda **GET** cât și metoda **POST** pot fi utilizate pentru descărcarea conținutului unei pagini Internet, transmițând către serverul web valorile unor anumite atribute, între acestea există anumite diferențe:

- o cerere **GET** poate fi reținută în cache, fapt ce nu este valabil și pentru o cerere **POST**;
- o cerere **GET** rămâne în istoricul aplicației de navigare, fapt ce nu este valabil și pentru o cerere **POST**;
- o cerere **GET** poate fi reținută printre paginile Internet favorite din cadrul programului de navigare, fapt ce nu este valabil și pentru o cerere **POST**;
- o cerere **GET** impune unele restricții cu privire la lungimea (maxim 2048 caractere) și la tipul de date (doar caractere) transmise (prin), fapt ce nu este valabil și pentru o cerere **POST**;
- o cerere **GET** nu trebuie folosită atunci când sunt implicate informații critice (acestea fiind vizibile în), fapt ce nu este valabil și pentru o cerere **POST**;
- o cerere **GET** ar trebui să fie folosită doar pentru obținerea unei resurse, fapt ce nu este valabil și pentru o cerere **POST**.

O linie de cerere HTTP poate fi succedată de unele informații suplimentare, reprezentând **antetele de cerere**, acestea având forma **atribut:valoare**, fiind definite următoarele proprietăți:

- **User-Agent** - informații cu privire la browser-ul utilizat și la platforma pe care rulează acesta
- informații cu privire la conținutul pe care clientul îl dorește de la serverul web, având capacitatea de a-l procesa; dacă serverul poate alege dintre mai multe resurse pe care le găzduiește, va alege pe cele care respectă constrângerile specificate, altfel întoarce un cod de eroare
 - **Accept** - tipul MIME
 - **Accept-Charset** - setul de caractere
 - **Accept-Encoding** - mecanismul de codificare
 - **Accept-Language** - limba
- **Host** (obligatoriu) - denumirea gazdei pe care se găsește resursa (specificată în); necesară întrucât o adresă IP poate fi asociată mai multor nume de
- **Authorization** - informații de autentificare în cazul unor operații care necesită drepturi privilegiate
- **Cookie** - transmite un cookie primit anterior
- **Date** - data și ora la care a fost transmisă cererea

Structura unui Răspuns HTTP

Un răspuns HTTP este format din linia de stare, antetele de răspuns și posibile informații suplimentare, conținând o parte sau toată resursa care a fost solicitată de client de pe serverul web.

În cadrul **liniei de stare** este inclus un cod din trei cifre care indică dacă solicitarea a putut fi îndeplinită sau nu (situație în care este indicată și cauza).

FAMILIE DE CODURI	SEMNIFICAȚIE	DESCRIERE
1xx	Informație	răspuns provizoriu, constând din linia de stare și alte antete (fără conținut, terminat de o linie vidă), indicând faptul că cererea a fost primită, procesarea sa fiind încă în desfășurare; nu este utilizată în HTTP/1.0
2xx	Succes	răspuns ce indică faptul că cererea a fost primită, înțeleasă, acceptată și procesată cu succes
3xx	Redirectare	răspuns transmis de serverul web ce indică faptul că trebuie realizate acțiuni suplimentare din partea clientului (cu sau fără interacțiunea utilizatorului, în funcție de metoda folosită) pentru ca cererea să poată fi îndeplinită; în cazul în care redirectarea se repetă de mai multe ori, se poate suspecta o buclă infinită

4xx	Eroare la client	răspuns transmis de serverul web ce indică faptul că cererea nu a putut fi îndeplinită, datorită unei erori la nivelul clientului; mesajul include și o entitate ce conține o descriere a situației, inclusiv tipul acesteia (permanentă sau temporară)
5xx	Eroare la server	cod de răspuns ce indică clientului faptul că cererea nu a putut fi îndeplinită, datorită unei erori la nivelul serverului web; mesajul include și o entitate ce conține o descriere a situației, inclusiv tipul acesteia (permanentă sau temporară)

Mesajul conține și **antetele de răspuns**, având forma **atribut :valoare**, fiind definite următoarele proprietăți:

- **Server** - informații cu privire la mașina care găzduiește resursa care este transmisă
- informații cu privire la proprietățile conținutului care este transmis
 - **Content-Encoding** - mecanismul de codificare
 - **Content-Language** - limba
 - **Content-Length** - dimensiunea
 - **Content-Type** - tipul MIME
- **Last-Modified** - ora și data la care pagina Internet a fost modificată
- **Location** - informație prin care serverul web informează clientul de faptul că ar trebui folosit alt (resursa a fost mutată sau trebuie accesată o pagină Internet localizată în funcție de anumite preferințe)
- **Accept-Ranges** - informație referitoare la transmiterea conținutului solicitat în mai multe părți, corespunzătoare unor intervale de octeți
- **Set-Cookie** - transmiterea unui cookie de la serverul web la client, acesta trebuind să fie inclus în antetele ulterioare ale mesajelor schimbate între cele două entități

Partea 2 : Proiectarea unui server web

Pregatirea spațiului de lucru

Pentru început creați un dosar Lab7 în spațiul personal pe calculatorul Dumneavoastră. El va conține în continuare fișierele cu paginile web, imaginile, etc. de pe site-ul Dumneavoastră. În versiune inițială serverul nu va fi capabil să lucreze cu imaginile și cu erorile. Aceste aspecte vor fi tratate în exercițiile complementare. Poziționați-vă în dosarul Lab7 și descarcați arhiva *siteWeb.tar.gz* de pe server

```
cd ~/Lab7
wget http://cs.fcim.utm.md/share/siteWeb.tar.gz
```

Decompresați arhiva în acest dosar.

```
tar -xvf siteWeb.tar.gz
```

Codul Dumneavoastră și fișierele binare se vor afla de asemenea în acest dosar.

Funcționarea serverului

În momentul conectării clientului la server (deschiderea unei conexiuni TCP), el va trimite o solicitare HTTP în formatul ASCII cu următoarea sintaxă (se cere pagina index.html la adresa de loopback)

```
GET /index.html HTTP/1.1\r\n
host: 127.0.0.1\r\n\r\n
```

Dacă nu este specificat numele paginii web serverul va trimite pagina implicită :

```
GET / HTTP/1.1\r\n
host: 127.0.0.1\r\n\r\n
```

Răspunsul serverului va depinde de mai mulți factori. Pentru început vom considera doar cazul normal, deci fără de erori. În acest caz se va trimite antetul HTTP pentru codul 200 urmat de pagina web cerută.

```
HTTP/1.1 200 OK\r\n
Content-Type: text/html\r\n\r\n
pagina web ceruta...
```

Exercițiul 1 : un server web simplu

Creați în limbajul C un server web care are să realizeze câteva funcții de bază prevăzute de protocolul http: ar primi cereri de la client și ar răspunde la ele

Pentru funcționarea corectă a serverului e necesară codarea următorilor pași :

1. la conectarea unui client trebuie sa recuperați numele fișierului cerut de către el ;
2. in continuare trebuie sa-i trimiteți codul HTTP indicat mai sus (codul 200) confirmând prin aceasta corectitudinea cererii;
3. trebuie sa-i trimiteți clientului fișierul cerut și sa închideți conexiunea

Pentru început trebuie sa citiți datele trimise de către client , lungimea șirului de caractere rezervat cererii trebuie sa fie de 1000 octeți. Pentru pasul 1 utilizați funcția ce se afla în fișierul *"parseRequest.txt"* pe care îl veți găsi în arhiva decompresată. Această funcție va folosi drept argument șirul de caractere ce provine de la client și va trimite în argumentul `string` numele fișierului cerut de el. Această funcție verifică de asemenea corectitudinea sintaxei cererii, în caz contrar șirul trimis este `NULL`. Funcția trimite 0 in caz de reușită și -1 în caz de eroare.

La pasul 2 este suficient să se scrie/trimită direct șirul de caractere indicat mai sus la socket-ul clientului. La pasul 3 trebuie sa deschideți fișierul cerut de client, sa-l citiți ai să trimiteți datele citite către client. După aceasta socket-ul clientului va fi închis.

Testați serverul în mod local (din mașina Linux) utilizând telnet (presupunem ca serverul ascultă pe portul 2000) :

```
telnet 127.0.0.1 2000
```

Introduceți manual cererea

```
GET /index.html HTTP/1.0  
host : 127.0.0.1:2000 <CRLF><CRLF>
```

Este important sa precizați după documentul cerut protocolul utilizat (aici se cere HTTP/1.0, dar ar fi putut fi și HTTP/1.1). Introduceți două `<CRLF>` (retur de linie) după comanda GET pentru a semnaliza serverului corectitudinea comenzii.

Testați în continuare serverul cu un navigator web de pe mașina gazda (dacă utilizați Virtualbox). Utilizați adresa IP a mașinii Linux și numele fișierului din dosarul Dumneavoastră, de exemplu :

```
http://192.168.100.1:2000/index.html
```

Testați de asemenea serverul conectându-vă de pe calculatorul unui coleg.

Exercițiul 2 : Gestionarea erorilor

In cazul producerii unei erori serverul trimite clientului un cod special (diferit de 200) și anumite pagini speciale. Vom considera doar două tipuri de erori : erori de sintaxă și erori când fișierul cerut nu există . In primul caz funcția `parseRequest()` trimite -1. Codul http și pagina web din arhiva `siteWeb.zip` care trebuie transmise clientului sunt :

```
HTTP/1.1 400 BAD REQUEST\r\n  
Content-Type: text/html\r\n\r\n  
Pagina web file400.html
```

În cel de-al doilea caz apelul la funcția pentru a deschide un fișier (open, fopen, etc.) transmite un cod de eroare. Codul http și pagina web din arhiva siteWeb.zip care trebuie transmise clientului sunt :

```
HTTP/1.1 404 FILE NOT FOUND\r\n
Content-Type: text/html\r\n\r\n
Pagina web file404.html
```

E posibilă o abordare mai fină a erorilor. Dacă folosiți open pentru a citi un fișier și are loc o eroare, vedeți valoarea errno. Ea vă va permite să cunoașteți mai precis tipul erorii (lipsa fișierului, permisiuni insuficiente, etc.). În principiu, pentru fiecare tip de eroare codul http va fi diferit (de exemplu : o problema de permisiuni va fi identificată cu codul 500 : internal server error).

Exercitiul 3 : Gestionarea imaginilor

La moment serverul lucrează doar cu fișiere textuale (html și txt). Acest exercițiu ne va permite să lucrăm și cu imagini. În cazul când serverul răspunde clientului, el indică și tipul fișierului trimis (Content-Type: text/html\r\n\r\n). Aceasta permite navigatorului să știe cum să interpreteze fișierul : o imagine va fi interpretată în mod diferit decât un fișier ASCII. Prin urmare, când un fișier transmis clientului nu este unul cu text, trebuie să fie indicat conținutul lui prin " Content-Type ". Pentru o imagine :

```
HTTP/1.1 200 OK\r\n
Content-Type: image/png\r\n\r\n
fișierul cu imagine...
```

În acest exercițiu se cere să analizați extensia fișierului cerut de către client. Dacă e vorba despre o imagine, transformați codul pentru ca serverul să transmită clientului codul corespunzător. Testați funcționarea serverului cu navigatorul web și cu una dintre imaginile care se afla în dosarul site-ului web.

Exercitiul 4 : Jurnalizarea

În momentul când un client se conectează puteți recupera adresa sa IP și URL-ul cerut (ceea ce și fac toate serverele web). Pentru aceasta trebuie să indicați în accept() un pointer la o structură sockaddr_in care va fi actualizat de către sistemul cu adresa IP a clientului.

Conținutul raportului

- Pagina cu titlu
- Scopul lucrării
- Descrierea codului
- Rezultatele experiențelor
- Concluzii

Depuneți raportul și o arhivă cu codul elaborat cel târziu după două săptămâni de la îndeplinirea lucrării.

Referințe

1. Radu-Lucian Lupsa "Eretele de calculatoare", capitolul 6 "Programarea în rețea – introducere", Casa Cărții de Știință, 2008
2. Michael J. Donahoo and Kenneth L. Calvert "TCP/IP Sockets in C. Practical Guide for Programmers", MKP, 2001
3. Ghidul Beej pentru programarea în rețea folosind socket de internet
<http://weknowyourdreams.com/beej.html>