Programare de sistem și de rețea

Lucrare de laborator nr.7

Proiectarea unui server web iterativ

Scopul lucrării

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

Partea 1 : Protocolul HTTP

Introducere

Un server şi un client web utilizează un protocol special care se numeşte HTTP (Hypertext Transfer Protocol) şi care este un protocol de comunicaţie responsabil cu transferul de hipertext (text structurat ce conţine legături) dintre un client (de regulă, un navigator) şi un server web, interacţiunea 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* şi *răspunsul*.

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

Structura unei cereri HTTP

O cerere HTTP conţine una sau mai multe linii de text , precedate în mod necesar de denumirea metodei specificând operaţia ce se va realiza asupra conţinutului 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	obţinerea 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 şi datele de autentificare, utilizatorul respectiv trebuind să aibă permisiunile necesare pentru o astfel de operatie

POST	transferul de informaţii 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	ştergerea resursei specificate de pe serverul web, rezultatul operaţiei depinzând de permisiunile pe care le deţine 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 distincţie între minuscule şi majuscule.

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

GET vs. POST

Deşi atât metoda GET cât şi metoda POST pot fi utilizate pentru descărcarea conţinutului unei pagini Internet, transmiţând către serverul web valorile unor anumite atribute, între acestea există anumite diferenţe:

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

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

- `User-Agent` - informaţii cu privire la browser-ul utilizat şi la platforma pe care rulează acesta
- informaţii cu privire la conţinutul pe care clientul îl doreşte de la serverul web, având capacitatea de a-l procesa; dacă serverul poate alege dintre mai multe resurse pe care le găzduieşte, 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 găseşte resursa (specificată în); necesară întrucât o adresă IP poate fi asociată mai multor nume de
- `Authorization` - informaţii de autentificare în cazul unor operaţii care necesită drepturi privilegiate
- `Cookie` - transmite un cookie primit anterior
- `Date` - data şi ora la care a fost transmisă cererea

Structura unui Răspuns HTTP

Un răspuns HTTP este format din linia de stare, antetele de răspuns şi posibile informaţii suplimentare, conţinând 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 (situaţie în care este indicată şi cauza).

FAMILIE DE CODURI	SEMNIFICAŢIE	DESCRIERE
1xx	Informaţie	răspuns provizoriu, constând din linia de stare şi alte antete (fără conţinut, terminat de o linie vidă), indicând faptul că cererea a fost primită, procesarea sa fiind încă în desfăşurare; nu este utilizată în HTTP/1.0
2xx	Succes	răspuns ce indică faptul că cererea a fost primită, înţeleasă, acceptată şi procesată cu succes
3xx	Redirectare	răspuns transmis de serverul web ce indică faptul că trebuie realizate acţiuni suplimentare din partea clientului (cu sau fără interacţiunea utilizatorului, în funcţie 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 şi o entitate ce conţine o descriere a situaţiei, 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 şi o entitate ce conţine o descriere a situaţiei, inclusiv tipul acesteia (permanentă sau temporară)

Mesajul conţine şi **antetele de răspuns**, având forma `atribut:valoare`, fiind definite următoarele proprietăţi:

- `Server` - informaţii cu privire la maşina care găzduieşte resursa care este transmisă
- informaţii cu privire la proprietăţile conţinutului care este transmis
    - `Content-Encoding` - mecanismul de codificare
    - `Content-Language` - limba
    - `Content-Length` - dimensiunea
    - `Content-Type` - tipul MIME
- `Last-Modified` - ora şi data la care pagina Internet a fost modificată
- `Location` - informaţie 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 funcţie de anumite preferinţe)
- `Accept-Ranges` - informaţie referitoare la transmiterea conţinutului solicitat în mai multe părţi, corespunzătoare unor intervale de octeţi
- `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 entităţi

Partea 2 : Proiectarea unui server web

Pregatirea spaţiului de lucru

Pentru început creaţi un dosar Lab7 în spaţiul personal pe calculatorul Dumneavoastră. El va conţine în continuare fişierele cu paginile web, imaginile, etc. de pe site-ul Dumneavoastră. În versiune iniţială serverul nu va fi capabil sa lucreze cu imaginile şi cu erorile. Aceste aspecte vor fi tratate in exerciţiile complementare. Poziţionaţi-vă in dosarul Lab7 şi descarcaţi arhiva *siteWeb.tar.gz* de pe server

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

Decompresaţi arhiva in acest dosar.

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

Codul Dumneavoastră şi fişierele binare se vor afla de asemenea in acest dosar.

Funcţionarea serverului

In momentul conectării clientului la server (deschiderea unei conexiuni TCP), el va trimite o solicitare HTTP in 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 mulţi factori. Pentru început vom considera doar cazul normal, deci fără de erori. In acest caz se va trimite antetul HTTP pentru codul 200 urmat de pagina web ceruta.

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

Exerciţiul 1 : un server web iterativ

Creaţi în limbajul C un server web care are realiza câteva funcţii de bază prevăzute de protocolul http: ar primi cereri de la client şi ar răspunde la ele

Pentru funcţionarea corectă a serverului e necesara codarea următorilor paşi :

1. la conectarea unui client trebuie sa recuperaţi numele fişierului cerut de către el ;
2. in continuare trebuie sa-i trimiteţi codul HTTP indicat mai sus (codul 200) confirmând prin aceasta corectitudinea cererii;
3. trebuie sa-i trimiteţi clientului fişierul cerut şi să închideţi conexiunea

Pentru început trebuie sa citiţi datele trimise de către client , lungimea şirului de caractere rezervat cererii trebuie sa fie de 1000 octeţi. Pentru pasul 1 utilizaţi funcţia ce se afla în fişierul *"parseRequest.txt"* pe care îl veţi găsi în arhiva decompresată. Această funcţie va folosi drept argument şirul de caractere ce provine de la client şi va trimite în argumentul `string` numele fişierului cerut de el. Această funcţie verifică de asemenea corectitudinea sintaxei cererii, în caz contrar şirul trimis este `NULL`. Funcţia trimite 0 in caz de reuşită şi -1 în caz de eroare.

La pasul 2 este suficient să se scrie/trimită direct şirul de caractere indicat mai sus la socket-ul clientului. La pasul 3 trebuie sa deschideţi fişierul cerut de client, sa-l citiţi şi să trimiteţi datele citite către client. După aceasta socket-ul clientului va fi închis.

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

```
telnet 127.0.0.1 2000
```

Introduceţi manual cererea

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

Este important sa precizaţi după documentul cerut protocolul utilizat (aici se cere HTTP/1.0, dar ar fi putut fi şi HTTP/1.1). Introduceţi două <CRLF> (retur de linie) după comanda GET pentru a semnala serverului completitudinea comenzii.

Testaţi în continuare serverul cu un navigator web de pe maşina gazda (dacă utilizaţi Virtualbox). Utilizaţi adresa IP a maşinii Linux şi numele fişierului din dosarul Dumneavoastră, de exemplu :

http://192.168.100.1:2000/index.html

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

Exerciţiul 2 : Gestionarea erorilor

In cazul producerii unei erori serverul trimite clientului un cod special (diferit de 200) şi anumite pagini speciale. Vom considera doar două tipuri de erori : erori de sintaxă şi erori când fişierul cerut un nu există . In primul caz funcţia `parseRequest()` trimite -1. Codul http şi 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

In cel de-al doilea caz apelul la funcţia pentru a deschide un fişier (open, fopen, etc.) transmite un cod de eroare. Codul http şi 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 fina a erorilor. Dacă folosiţi `open` pentru a citi un fişier şi are loc o eroare, vedeţi valoarea `errno`. Ea vă va permite sa cunoaşteţi mai precis tipul erorii (lipsa fişierului, 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`).

Exercitilul 3 : Gestionarea imaginilor

La moment serverul lucrează doar cu fişiere textuale (html şi txt). Acest exerciţiu ne va permite sa lucrăm şi cu imagini. În cazul când serverul răspunde clientului, el indică şi tipul fişierului trimis (`Content-Type: text/html\r\n\r\n`). Aceasta permite navigatorului să ştie cum să interpreteze fişierul : o imagine va fi interpretată în mod diferit decât un fişier ASCII. Prin urmare, când un fişier transmis clientului nu este unul cu text, trebuie să fie indicat conţinutul lui prin " `Content-Type` ". Pentru o imagine :

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

In acest exerciţiu se cere să analizaţi extensia fişierului cerut de către client. Dacă e vorba despre o imagine, transformaţi codul pentru ca serverul să transmită clientului codul corespunzător. Testaţi funcţionarea serverului cu navigatorul web şi cu una dintre imaginile care se afla in dosarul site-ului web.

Exercitiul 4 : Jurnalizarea

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

Conţinutul raportului

- Pagina cu titlu
- Scopul lucrării
- Descrierea codului
- Rezultatele experienţelor
- 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 Cărţii de Ştiinţă, 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 in rețea folosind socket de internet http://weknowyourdreams.com/beej.html