4. Aplicații de rețea I (HTTP)

4.1. Cuprins modul

Introducere

Un nou domeniu public şi aplicații Internet proprietare sunt dezvoltate în fiecare zi. În loc să tratăm un număr mare de aplicații Internet, într-un mod enciclopedic, am ales să ne concentrăm pe un număr mic de aplicații, mai importante şi mai cunoscute.

Vom discuta, în primul rând, despre Web, nu doar pentru că este cea mai cunoscută aplicație, ci şi datorită faptului că protocolul său pentru nivelul aplicație, HTTP, este relativ simplu, funcționează deasupra TCP şi ilustrează câteva principii cheie ale protocoalelor de rețea.

Obiective

După parcurgerea acestei unități de curs studenții vor fi capabili:

- ✓ Să definească terminologia Web
- ✓ Să compare conexiunile HTTP nepersistente cu conexiunile HTTP persistente
- ✓ Să descrie structura mesajelor HTTP de cerere şi răspuns

Durată medie de studiu individual

Durata medie de studiu individual : 2 ore

4.2. World Wide Web: HTTP

În anii ’80, Internet-ul era folosit de către cercetători, studenți etc. (mediul universitar-academic-cercetare) pentru a se conecta la stații gazdă aflate la distanță, cu scopul de a transfera fişiere de la o gazdă locală la una aflată la distantă şi vice versa, pentru a primi şi trimite ştiri şi pentru a primi sau trimite mesaje prin poşta electronică. Cu toate că aceste aplicații erau (şi continuă să fie) foarte utile, Internet-ul nu era cunoscut în afara comunităților de academicieni şi cercetători. La începutul anilor ’90 a apărut cea mai importantă aplicație a Internet-ului – World Wide Web (“pânză de păianjen ce acoperă întreaga lume”). Web-ul este aplicația care a atras cel mai mult atenția publicului. Web-ul schimbă dramatic modul în care oamenii interacționează, unii cu alții, în interiorul/exteriorul mediului de lucru.

Ceea ce atrage atenția cel mai mult utilizatorilor la Web, este faptul că este *la cerere* (eng. on demand). Utilizatorii primesc ce vor, când vor, Web-ul fiind diferit de difuzarea radio/televiziune, unde utilizatorii sunt forțați să preia conținutul atunci când furnizorul îl face disponibil. Este foarte uşor pentru cineva să facă public un lucru, prin intermediul Web-ului; oricine poate deveni un autor, la un preț foarte mic. Hiperlegăturile şi motoarele de căutare ne ajută să navigăm într-un ocean de site-uri Web. Formulare, applet-uri Java, componente Active X, plus multe alte tehnologii, ne permit să interacționăm cu paginile şi site-urile. Şi, din ce în ce mai mult, Web-ul implementează o interfață meniu, către cantități mari de materiale audio/video, stocate în Internet, audio şi video care pot fi accesate la cerere.

4.3. Privire de ansamblu asupra HTTP

HTTP (HyperText Transfer Protocol), protocolul nivelului aplicație pentru Web, este nucleul Web-ului. HTTP este implementat în două programe: un program client şi un program server. Programele client şi server, executate pe sisteme capăt diferite, comunică între ele printr-un schimb de mesaje HTTP. HTTP defineşte structura acestor mesaje şi modul în care clientul şi server-ul schimbă aceste mesaje între ei. Înainte de a intra în detalii, este important să recapitulăm terminologia Web.

Pagina Web (denumită şi document) constă din obiecte. Un *obiect* este un fişier simplu (fişier HTML, imagine JPEG, imagine GIF, applet Java, clip audio etc), care este adresabil printr-un singur URL (Universal Resource Locator).

Majoritatea paginilor Web constau dintr-un *fişier HTML de bază* şi câteva obiecte referite. De exemplu, dacă o pagină Web conține text HTML şi cinci imagini JPEG, atunci pagina Web conține şase obiecte: fişierul HTML de bază plus cele cinci imagini. Fişierul HTML de bază face referință la celelalte obiecte din pagină, prin URL-urile obiectelor. Fiecare URL are trei componente: numele protocolului, numele stației gazdă, de la server-ul care găzduieşte obiectul şi

numele căii obiectului. De exemplu, URL-ul: http://www.unitbv.ro/iesc/sigla.jpg are numele protocolului *http*, numele stației gazdă *www.unitbv.ro* şi numele căii */iesc/sigla.jpg*. Atunci când nu este specificat protocolul vom considera implicit că el este http.

Navigatorul este un agent utilizator pentru Web; el afişează utilizatorului pagina cerută şi pune la dispoziție numeroase caracteristici de navigare şi de configurare. Navigatoarele Web implementează, de asemenea partea de client pentru HTTP. De aceea, în contextul Web-ului, vom folosi termenii “navigator” şi “client” alternativ. Navigatoarele Web populare includ Mozilla Firefox şi MS Internet Explorer.

Server-ul Web găzduieşte obiecte Web, fiecare obiect fiind adresabil prin URL. Server-ul Web implementează, de asemenea, partea de server pentru HTTP. Serverele Web populare includ Apache şi Microsoft Internet Information Server.

HTTP defineşte modul în care clienții Web (ex. navigatoare) cer pagini Web de la servere (ex. servere Web) şi modul în care server-ele transferă pagini Web la clienți. Vom analiza interacțiunea dintre client şi server, puțin mai jos, însă ideea generală este ilustrată în Figura 4.1. Atunci când un utilizator cere o pagină Web (ex. face clic pe o hiperlegătura), navigatorul trimite mesaje HTTP de cerere către server, pentru obiectele din pagină. Server-ul primeşte cererile şi răspunde prin mesaje HTTP de răspuns, care conțin obiectul. Până in anul 1997, toate navigatoarele şi server-ele Web implementau versiunea HTTP/1.0, definită în [RFC 1945]. Începând cu anul 1998, server-ele Web şi navigatoarele au început să implementeze versiunea HTTP/1.1, definită în [RFC 2068]. HTTP/1.1 este compatibil cu HTTP/1.0; un server Web ce rulează pe 1.1, poate comunica cu un navigator ce rulează 1.0, iar un navigator ce rulează 1.1, poate comunica cu un server ce rulează versiunea 1.0.

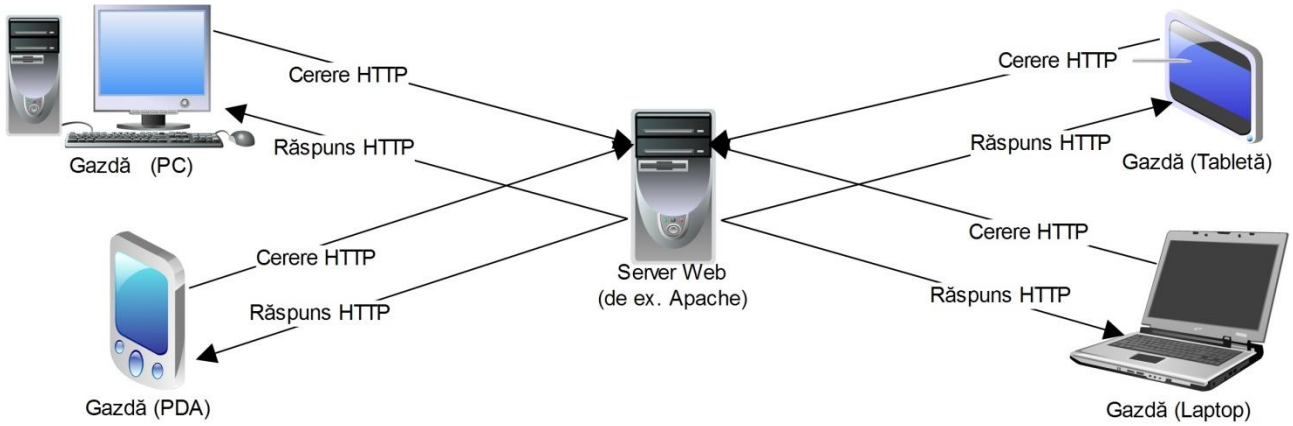


Figura 4.1 Comportamentul HTTP cerere-răspus

Atât HTTP/1.0 cât şi HTTP/1.1 utilizează TCP ca şi protocol de transport de bază (în loc să ruleze deasupra UDP). Prima dată, clientul HTTP inițiază o conexiune TCP cu server-ul. Odată stabilită conexiunea, procesele navigatorului şi server-ului accesează TCP, prin intermediul

interfețelor soclurilor. Clientul trimite mesaje HTTP de cerere, în interfața soclului său şi primeşte mesaje HTTP de răspuns de la interfața soclului său. În mod similar, server-ul HTTP primeşte mesaje de cerere, de la interfața soclului său, şi trimite mesaje de răspuns, în interfața soclului său. Odată ce clientul trimite mesaje în interfața soclului său, mesajul nu se mai află "în grija" clientului, ci se află "în grija" TCP-ului. Să presupunem că utilizatorul doreşte să vizualizeze pagina www.unitbv.ro/iesc/index.html. Redăm secvența de paşi executată de navigator:

1. navigatorul interoghează DNS-ul asupra adresei IP a www.unitbv.ro;

2. DNS-ul răspunde cu 193.254.231.8;

3. Navigatorul realizează conexiunea TCP cu portul 80 al gazdei 193.254.231.8;

4. Se trimite o cerere HTTP pentru documentul /iesc/index.html;

5. Serverul www.unitbv.ro furnizează un răspuns cu fişierul index.html din dosarul iesc;

6. În funcție de capabilitățile server-ului şi clientului, conexiunea este eliberată sau menținută pentru o nouă cerere;

7. Navigatorul afişează textul din /iesc/index.html;

8. Navigatorul solicită şi afişează obiectele înglobate în documentul de bază.

Multe programe de navigare informează utilizatorul asupra etapei în care se află prin intermediul unei bare de stare aflate în partea de jos a ferestrei programului. În acest fel, dacă performanțele sunt sub aşteptări utilizatorul poate afla dacă server-ul DNS nu răspunde, server-ul Web nu răspunde sau rețeaua este congestionată pe parcursul transmiterii răspunsului HTTP.

În scopul furnizării unui răspuns server-ul execută următorii paşi:

1. Acceptă o conexiune TCP de la client

2. Extrage numele fişierului din mesajul HTTP de cerere

3. Citeşte fişierul de pe disc (dacă există)

4. Trimite clientului mesajul de răspuns ce conține fişierul solicitat (dacă acesta există, dacă nu trimite un mesaj de eroare)

5. În funcție de capabilitățile serverului şi clientului conexiunea poate fi eliberată sau menținută pentru a răspunde la o nouă cerere.

Să ne amintim faptul că TCP pune la dispoziție HTTP-ului un serviciu de transfer de date fiabil. Aceasta implică faptul că, fiecare mesaj de cerere HTTP, emis de un proces client, va ajunge intact la server; în mod similar, fiecare mesaj de răspuns HTTP, emis de către procesul server, va ajunge intact la client. Aici vedem unul din marile avantaje ale arhitecturii pe nivele – HTTP nu trebuie să aibă grija datelor pierdute, sau detalii despre cum TCP recuperează datele pierdute sau reordonează datele în rețea. Aceasta este sarcina TCP-ului şi a protocoalelor din nivelele inferioare, din stiva de protocoale.

Este important de remarcat că server-ul trimite fişierele cerute către clienți, fără a stoca informații de stare despre client. Dacă un anumit client cere acelaşi obiect, de două ori într-o perioadă de câteva secunde, server-ul nu-i va răspunde acestuia că abia a transmis acel obiect; server-ul va retrimite obiectul, ca şi cum ar fi uitat operația executată anterior. Datorită faptului că HTTP nu menține informații despre clienți, se spune că este un protocol *fără stări* (eng. stateless).

4.4. Conexiuni non-persistente şi persistente

HTTP poate utiliza atât conexiuni persistente cât şi non-persistente. Conexiunile non-persistente reprezintă modul implicit pentru HTTP/1.0, iar conexiunile persistente pentru HTTP/1.1.

4.4.1. Conexiuni non-persistente

Să parcurgem paşii transferului unei pagini Web de la server la client, pentru cazul unei conexiuni non-persistente. Să presupunem că pagina constă dintr-un fisier HTML de bază şi 10 imagini JPEG, şi toate cele 11 obiecte sunt pe acelaşi server. Presupunem că URL-ul pentru fişierul HTML de bază este www.unitbv.ro/iesc/index.html. Să vedem ce se întâmplă:

1. Clientul HTTP inițiază o conexiune TCP către server-ul www.unitbv.ro. Numărul portului este 80, folosit ca număr de port implicit, la care server-ul HTTP va asculta clienții HTTP, care vor să retragă documente utilizând HTTP.
2. Clientul HTTP trimite un mesaj de cerere HTTP în soclul asociat conexiunii TCP, stabilită în pasul anterior. Mesajul de cerere va include ori întregul URL, ori doar numele căii /iesc/index.html . Vom discuta mai detaliat despre mesajele HTTP în secțiunile următoare.
3. Server-ul HTTP primeşte mesajul de cerere prin soclul asociat conexiunii stabilite la pasul 1, retrage obiectul /iesc/index.html din locul în care este stocat (RAM sau disc), încapsulează obiectul într-un mesaj de răspuns HTTP şi trimite mesajul de răspuns în conexiunea TCP.
4. Server-ul HTTP îi spune TCP-ului să închidă conexiunea TCP. (Dar TCP nu va încheia conexiunea până când clientul nu primeşte mesajul de răspuns intact.)

5. Clientul HTTP primeşte mesajul de răspuns. Conexiunea TCP se încheie. Mesajul indică faptul că obiectul încapsulat este un fişier HTML. Clientul extrage fişierul din mesajul de răspuns, analizează fişierul HTML şi găseşte referințele la cele 10 obiecte JPEG.
6. După aceea se repetă primii patru paşi pentru fiecare din obiectele JPEG referite.

În momentul în care navigatorul primeşte pagina Web, acesta o va afişa utilizatorului. Două navigatoare diferite pot interpreta (ex. afişare către utilizator) o pagină Web în două moduri oarecum diferite. HTTP nu tratează modul în care clientul interpretează o pagină Web. Specificațiile HTTP ([RFC 1945] şi [RFC 2068]) definesc doar protocolul de comunicație dintre programul client HTTP şi programul server HTTP.

Paşii de mai sus utilizează conexiuni non-persistente deoarece, fiecare conexiune TCP este închisă după ce server-ul trimite obiectul – conexiunea nu persistă pentru alte obiecte. Astfel fiecare conexiune TCP transportă exact un mesaj de cerere şi un mesaj de răspuns. Aşadar, în acest exemplu, când un utilizator cere pagina Web, se generează 11 conexiuni TCP.

În paşii descrişi mai sus, am vorbit intenționat mai vag despre întrebarea dacă respectivul client obține cele 10 imagini JPEG pe 10 conexiuni seriale TCP, sau dacă unele din imaginile JPEG sunt obținute pe conexiuni paralele TCP. Într-adevăr, utilizatorii pot configura navigatoarele moderne să controleze gradul de paralelism. În modurile lor implicite, majoritatea navigatoarelor deschid 5 până la 10 conexiuni paralele TCP, şi fiecare din aceste conexiuni se ocupă de câte o tranzacție cerere-răspuns. Dacă utilizatorul preferă, poate seta numărul maxim de conexiuni paralele la una iar în acest caz cele 10 conexiuni sunt stabilite serial. După cum vom vedea în capitolul următor, utilizarea conexiunilor paralele scurtează timpul de răspuns, întrucât sunt eliminate unele din întârzierile dus-întors (RTT) şi start lent de la TCP.

Înainte de a continua, să facem un calcul pentru a estima intervalul de timp din momentul în care un client cere fişierul de bază HTML până în momentul în care fişierul este primit de către client. *Timpul dus-intors (RTT, eng. Roud Trip Time)* reprezintă timpul necesar unui pachet mic, pentru a călători de la client la server şi apoi, înapoi la client. RTT include întârzierile de propagare ale pachetelor, întârzierile in coadă ale pachetului pe rutere-le şi switch-urile intermediare, şi întârzierile de procesare ale pachetului. Acum, să considerăm situația în care un utilizator face clic pe o hiperlegătură. Aceasta face ca navigatorul să inițieze o conexiune TCP între navigator şi server-ul Web; aceasta implică o înțelegere in trei paşi specifica TCP-ului – clientul trimite un mesaj TCP mic către server, server-ul confirmă şi răspunde cu un mesaj mic, şi în final clientul îi confirmă server-ului. Un RTT trece după primele două etape ale înțelegerii. După completarea primelor două etape ale înțelegerii, clientul trimite mesajul de cerere HTTP în conexiunea TCP, iar TCP ataşează ultima confirmare (a treia etapă din înțelegerea in trei paşi) la mesajul de cerere. Odată ce mesajul de cerere ajunge la server, server-ul trimite fişierul HTML în conexiunea TCP.

Acest ciclu cerere/răspuns HTTP face să mai treacă încă un RTT. Aşadar, în mare, timpul total de răspuns este 2RTT, la care se adaugă timpul de transmisie a fişierului HTML la server.

4.4.2. Conexiuni persistente

Conexiunile non-persistente au anumite dezavantaje.

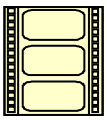
- În primul rând, o nouă conexiune trebuie stabilită şi menținută pentru fiecare obiect cerut. Pentru fiecare din aceste conexiuni, trebuie alocat spațiu de memorie pentru TCP, iar variabilele TCP trebuiesc memorate atât în client cât şi în server. Aceasta poate face ca supraîncărcare (necesar de memorie RAM, timp de procesare) la server-ul Web să devină foarte mare în momentul servirii simultane.
- În al doilea rând, după cum abia am descris, fiecare obiect necesită 2RTT – un RTT pentru a stabili conexiunea TCP şi un RTT pentru a cere şi primi un obiect.
- În ultimul rând, fiecare obiect este afectat de startul lent de la TCP, întrucât fiecare conexiune TCP începe cu o fază de start lent. Însă, acumularea RTT şi întârzierile de start lent sunt parțial înlăturate prin folosirea conexiunilor TCP paralele.

Cu conexiunile persistente, server-ul lasă conexiunea TCP deschisă după trimiterea răspunsurilor. Cererile şi răspunsurile ulterioare între acelaşi client şi server pot fi trimise pe aceeaşi conexiune. În particular, o pagină Web întreagă (în exemplul de mai sus, fişierul de bază HTML şi cele 10 imagini) poate fi trimisă pe o conexiune TCP persistentă; mai mult, pagini Web multiple, care se regăsesc pe acelaşi server, pot fi trimise pe o conexiune TCP. Tipic, server-ul HTTP închide conexiunea atunci când nu este folosită un anumit interval de timp (intervalul de pauză), care este adesea configurabil.

Există două tipuri de legături persistente: *cu bandă de asamblare* sau *fără bandă de asamblare*. Pentru versiunea *fără bandă de asamblare*, clientul face o nouă cerere doar atunci când răspunsul anterior a fost primit. În acest caz, fiecare din obiectele referite (cele 10 imagini din exemplul de mai sus) are nevoie de un RTT pentru a cere şi primi obiectul. Chiar dacă este o îmbunătățire față de cele două RTT de la conexiunile non-persistente, întârzierea RTT poate fi redusă mai departe utilizarea *benzii de asamblare*. Un alt dezavantaj al lipsei benzii de asamblare este că, după ce server-ul trimite un obiect prin conexiunea TCP persistentă, conexiunea stă pur şi simplu – nu face nimic – în timp ce aşteaptă să vină o altă cerere. Această aşteptare risipeşte din resursele server-ului.

Modul implicit pentru HTTP/1.1 utilizează conexiunile persistente cu bandă de asamblare. În acest caz, clientul HTTP inițiază o cerere în momentul în care întâlneşte o referință. Deci, clientul HTTP poate face cereri una după alta (back-to-back) pentru obiectele referite. În momentul

în care server-ul primeşte cererile, poate trimite obiectele unul după altul. Dacă toate cererile şi răspunsurile sunt trimise unul după altul, atunci se consumă doar un RTT pentru toate obiectele referite (decât un RTT pentru fiecare obiect referit, când nu se foloseşte banda de asamblare). Mai departe, conexiunea TCP folosită cu bandă de asamblare rămâne nefolosită pentru un interval mai mic de timp. În plus la conexiunile persistente (cu sau fără pipelining) întârzierea de start lent apare o singură dată.



Estimarea întârzierilor HTTP

http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/http/http.html

Simulare

4.5. Formatul mesajelor HTTP

Formatul mesajelor HTTP este specificat în [RFC 1945] pentru HTTP 1.0 şi în [RFC 2068] pentru versiunea 1.1 a protocolului. Există două tipuri de mesaje HTTP, mesaje de cerere şi mesaje de răspuns, ambele urmând a fi analizate mai jos.

4.5.1. Mesajul HTTP de cerere

Mai jos este descris un mesaj tipic de cerere HTTP:

Exemplu

```
GET /iesc/index.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: ro
```

Putem deduce foarte multe lucruri doar privind acest mesaj simplu de cerere. Înainte de toate, vedem că mesajul este scris în codul ASCII (text obişnuit), astfel încât orice persoană cu cunoştințe despre calculatoare să îl poată citi. În al doilea rând, vedem că mesajul constă din 5 linii, fiecare fiind urmată de un Carriage Return şi un Line Feed (mai simplu spus: Enter şi de la capăt). Ultima linie este urmată de un CR şi LF adiționale. Chiar dacă acest mesaj particular de cerere conține 5 linii, un mesaj de cerere poate avea fie mai multe linii, fie doar una singură. Prima linie a unui mesaj de cerere HTTP se numeşte linia de cerere şi constă din 3 câmpuri: *metoda*, *URL* şi *versiunea HTTP*. Câmpul metodă poate lua diferite valori, incluzând *GET, POST* şi *HEAD*.

Majoritatea mesajelor HTTP de cerere utilizează metoda GET. Metoda GET se foloseşte atunci când navigatorul cere un obiect, iar acest obiect e identificat în câmpul URL. În acest exemplu, navigatorul cere obiectul /iesc/index.html. (Navigatorul nu trebuie să specifice numele

stației gazdă în câmpul URL, de vreme ce conexiunea TCP este deja stabilită cu serverul care serveşte fişierul cerut.) Versiunea se explică de la sine; în acest exemplu, navigatorul implementează versiunea HTTP/1.1.

În continuare, vom analiza liniile mesajului de cerere exemplificat:

- Prin includerea liniei antet *Connection: close*, navigatorul îi spune server-ului că nu vrea să folosească conexiunile persistente; acesta vrea ca server-ul să închidă conexiunea după trimiterea obiectului cerut. Astfel navigatorul care a generat acest mesaj de cerere implementează HTTP/1.1, dar nu vrea să aibă de-a face cu conexiunile persistente.
- Linia antet *User-agent:* specifică agentul utilizator, ex., tipul navigatorului care face cererea către server. Aici agentul utilizator este Mozilla/4.0, un navigator Netscape. Această linie antet este utilă deoarece server-ul poate trimite versiuni diferite ale aceluiaşi obiect la diferite tipuri de agenți utilizator. (Fiecare din aceste versiuni este adresată prin acelaşi URL).
- Linia antet *Accept:* îi spune server-ului ce tipuri de obiecte este pregătit navigatorul să accepte. În acest caz, clientul este pregătit să accepte text HTML, imagini GIF sau imagini JPEG. Dacă fişierul /iesc/index.html conține un applet Java, atunci server-ul ar trebui să nu trimită fişierul, pentru că navigatorul nu poate interpreta acel tip de obiect.
- În cele din urmă, linia antet *Accept-language:* indică faptul că utilizatorul preferă să primească o versiune în limba romană a obiectului, dacă un astfel de obiect există pe server; altfel, server-ul ar trebui să trimită versiunea sa implicită.

Formatul general al unui mesaj de cerere este ilustrat în Figura 4.2

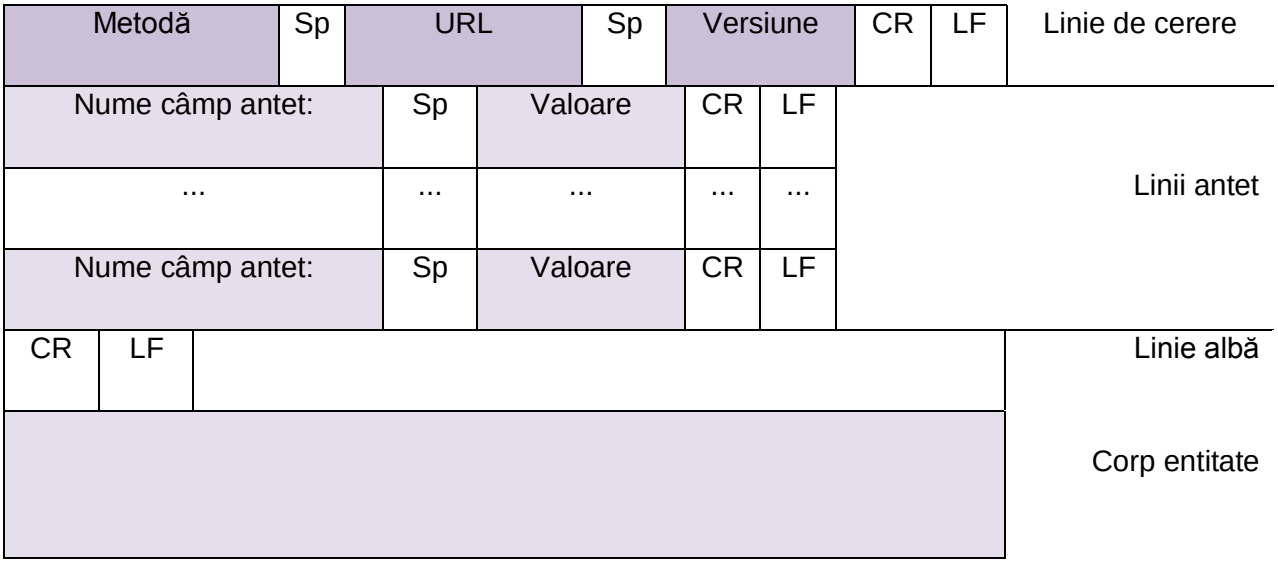


Figura 4.2 Formatul general al unui mesaj HTTP de cerere

După cum putem vedea, formatul general urmează îndeaproape mesajul de cerere din exemplul de mai sus. Poate ați observat, oarecum că, după liniile antet (şi CR şi LF adiționale), există un corp entitate. Corpul entitate nu e folosit cu metoda GET, dar e folosit cu metoda POST. Clientul HTTP utilizează metoda POST, atunci când utilizatorul completează un formular. Cu un mesaj POST, utilizatorul cere o pagină Web de la server, dar conținutul specific din pagina Web depinde de ce a scris utilizatorul în câmpurile formularului. Dacă valoarea din câmpul metodă este POST, atunci corpul entitate conține ceea ce utilizatorul a tastat în câmpurile formularului (plus alte informații).

Metoda HEAD este asemănătoare cu metoda GET. Atunci când un server primeşte o cerere cu metoda HEAD, el răspunde cu un mesaj HTTP, dar abandonează obiectul cerut, furnizând doar antetul HTTP. Metoda HEAD este adesea folosită de către dezvoltatorii server-elor HTTP, pentru depanare (eng. debugging).

4.5.2. Mesajul HTTP de răspuns

Mai jos este arătat un mesaj tipic de răspuns HTTP. Acest mesaj de răspuns ar putea fi răspunsul la mesajul de cerere, explicat anterior.

Exemple

```
HTTP/1.1 200 OK
Connection: close
Date: Mon, 06 Aug 2012 15:00:12 GMT
Server: Apache/2.3.0 (Unix)
Last-Modified: Fri, 03 aug 2012 09:33:24 GMT
Content-Length: 8635
Content-Type: text/html

data data data data data ........
```

Să privim mai atent acest mesaj de răspuns. El cuprinde trei secțiuni: o linie inițială de stare, 6 linii de antet şi apoi corpul entitate. Corpul entitate reprezintă informația mesajului – conține obiectul cerut (reprezentat de data data data data data.....). Linia de stare are trei câmpuri: *versiunea protocolului*, un *cod de stare* şi un *mesaj de stare* corespunzător. În acest exemplu, linia de stare indică faptul că server-ul utilizează HTTP/1.1 şi că totul este în ordine (ex. server-ul a găsit şi trimite obiectul cerut). Acum să analizăm liniile de antet:

- Server-ul utilizează *Connection:close*, pentru a-i spune clientului că va închide conexiunea TCP după ce va trimite mesajul.

- Date: indică timpul şi data când a fost creat şi trimis răspunsul HTTP de către server. De notat că acesta nu este timpul când obiectul a fost creat sau modificat ultima dată; este timpul când server-ul retrage obiectul din sistemul său de fişiere, introduce fişierul în mesajul de răspuns şi trimite mesajul de răspuns.
- Server: indică faptul că mesajul a fost generat de un server Apache Web; este analog cu linia de antet User-agent din mesajul de cerere HTTP.
- Last-Modified: este critică pentru stocarea obiectelor, atât la clientul local cât şi pe server-ele de rețea (denumite proxy).
- Content-Length: indică numărul de octeți din obiect, care sunt trimişi.
- Content-Type: indică faptul că obiectul din corpul entitate este text HTML. (Tipul obiectului este oficial indicat prin antetul Content-Type şi nu de extensia fişierului).

Observăm că dacă server-ul primeşte o cerere HTTP/1.0, nu va utiliza o conexiune persistentă, chiar dacă este un server HTTP/1.1. În schimb, server-ul HTTP/1.1 va închide conexiunea TCP după trimiterea obiectului. Aceasta este necesară pentru că un client HTTP/1.0 aşteaptă ca server-ul să închidă conexiunea.

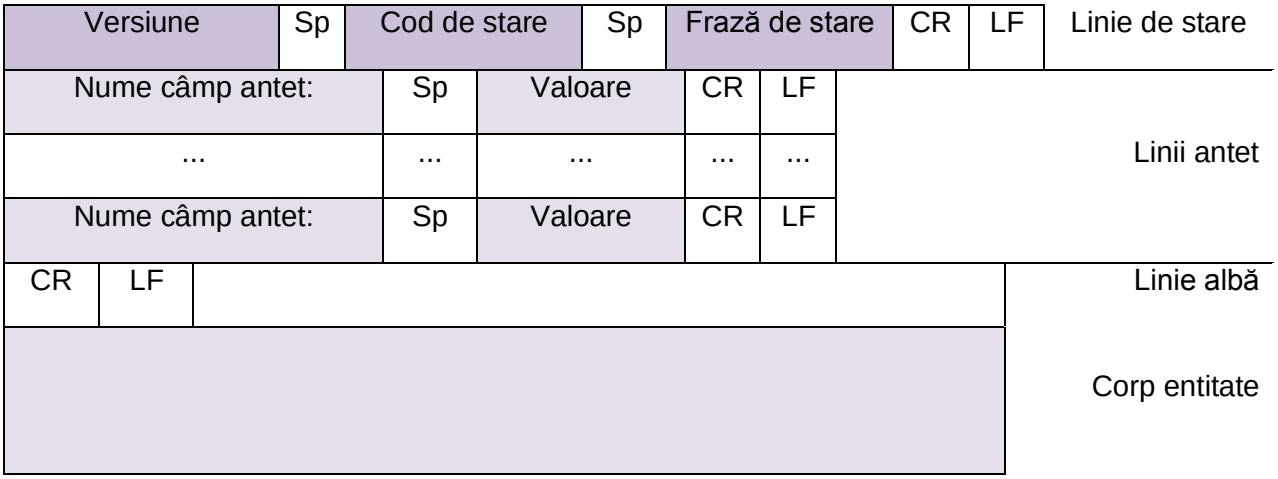


Figura 4.3 Formatul general al unui mesaj de răspuns

După ce am văzut un exemplu, să examinăm acum formatul general al unui mesaj de răspuns, arătat în figura 2.2-3. Acest format general al mesajului de răspuns se potriveşte cu cel din exemplul anterior, al unui mesaj de răspuns. Să mai adăugăm câteva cuvinte despre codurile de stare şi frazele lor. Codul de stare şi fraza asociată indică rezultatul cererii. Câteva coduri de stare comune şi frazele asociate includ:

Cod de stare	Fraza de stare	Semnificație
200	OK	Cererea a fost un succes şi informația a fost returnată în răspuns.

400	Bad Request	Un cod generic de eroare ce indică faptul că cererea nu a fost înțeleasă de către server.
404	Not Found	Documentul cerut nu există pe server

În această secțiune am discutat despre un număr de linii de antet, care pot fi folosite în mesajele HTTP de cerere şi răspuns. Specificația HTTP (în special HTTP/1.1) defineşte multe alte linii de antet ce pot fi introduse de către navigatoare, servere Web şi servere de rețea de stocare. Noi am acoperit doar o fracțiune din totalitatea liniilor de antet.

Cum decide un navigator ce linii de antet include într-un mesaj de cerere? Cum decide un server Web ce linii de antet include într-un mesaj de răspuns? Un navigator va genera linii de antet, ca o funcție a tipului şi versiunii de navigator (ex. un navigator HTTP/1.0 nu va genera linii de antet ce aparțin unui navigator HTTP/1.1), configurații utilizator ale navigatorului (ex. limba preferată) şi dacă navigatorul are, în mod curent, o versiune stocată, probabil expirată, a obiectului. Server-ele Web se comportă într-un mod asemănător: există diferite produse, versiuni şi configurații, toate influențând tipul liniilor de antet incluse în mesajul de răspuns.

Teste de autoevaluare

1. Adevărat sau fals

a. Un utilizator solicită o pagină Web ce conține un text şi trei imagini. Pentru această pagină clientul va trimite un mesaj de cerere şi va primi patru mesaje de răspuns.

b. Două pagini Web distincte (cum ar fi iesc.unitbv.ro/admitere.html şi iesc.unitbv.ro/studenti.html) pot fi trimise peste aceeaşi conexiune persistentă.

c. În cazul unei conexiuni nepersistente între navigator şi server, este posibil ca un singur segment TCP să transporte două mesaje HTTP de cerere distincte.

d. Antetul `Date:` din mesajul HTTP de răspuns indică când a fost modificat ultima dată obiectul din răspuns.

e. Mesajele HTTP de răspuns nu au niciodată un corp vid.

2. Ce se înțelege prin afirmația: HTTP este un protocol fără stări.

4.6. Memoria ascunsă Web (eng. Web Caching)

O memorie ascunsă Web – denumită şi server proxy – este o entitate de rețea care satisface cereri HTTP în locul unui server Web original. Web cache-ul dispune de propriul spațiu de stocare pe disc unde păstrează copii ale obiectelor cerute recent. Aşa cum este indicat în Figura XXX, navigatorul poate fi configurat astfel încât toate cererile HTTP să fie direcționate către Web cache. Ca exemplu să presupunem că navigatorul solicită obiectul: http://iesc.unitbv.ro-sigla.jpg. Iată ce se întâmplă:

1. Navigatorul stabileşte o conexiune TCP cu Web cache-ul şi trimite o cerere HTTP pentru obiect către Web cache;
2. Web cache-ul verifică dacă există o copie a obiectului stocată local. Dacă da, atunci returnează obiectul în cadrul unui mesaj HTTP de răspuns către navigator.
3. Dacă Web cache-ul nu dispune de acel obiect, atunci acesta deschide o conexiune TCP către serverul original, iesc.unitbv.ro în cazul nostru. Web cache-ul trimite o cerere HTTP pentru acel obiect folosind conexiunea TCP cache-server. După primirea cererii, serverul original furnizează obiectul în cadrul unui mesaj HTTP de răspuns adresat Web cache-ului;
4. La recepționarea obiectului, Web cache-ul îl stochează local şi trimite o copie în cadrul unui mesaj HTTP de răspuns către navigator (prin conexiunea TCP existentă între Web cache şi navigator)

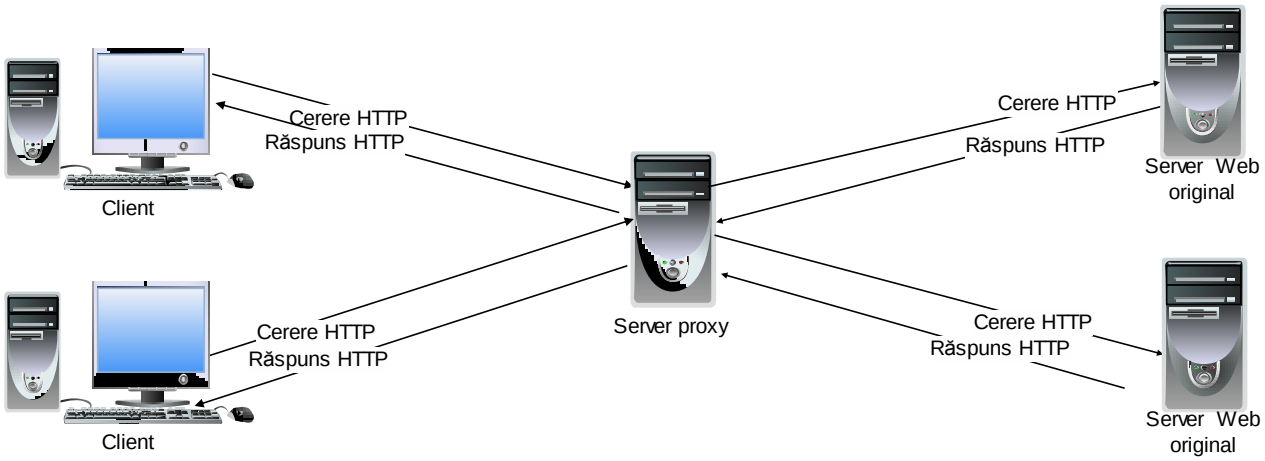


Figura 4.4 Clienți care solicit obiecte printr-un server proxy

Remarcăm faptul că Web cache-ul este simultan client şi server HTTP: atunci când primeşte cereri şi furnizează răspunsuri către navigator se comportă ca un server; atunci când trimite cereri şi recepționează răspunsuri de la serverele web originale se comportă ca un client.

De multe ori Web cache-urile sunt achiziționate şi instalate de către ISP-uri. Spre exemplu o universitate ar putea instala un cache în rețeaua de campus urmând să configurere toate navigatoarele pentru utilizarea acelui cache. Sau un furnizor de servicii internet rezidențial (cum ar fi rcs-rds) ar putea instala unul sau mai multe cach-uri în rețeaua sa forțând navigatoarele să-l utilizeze.

Web cache-ul a fost dezvoltat din două motive. În primul rând, un Web cache poate reduce substanțial timpul de răspuns pentru o cerere venită din partea unui client, în special dacă lărgimea de bandă dintre client şi serverul Wen original este mai mică decât lărgimea de bandă dintre client şi Web cache. Dacă există o conexiune de mare viteză între client şi cache şi cache-ul dispune de obiectul cerut, atunci cache-ul va fi capabil să furnizeze obiectul rapid către client. În al doilea rând, aşa cum vom vedea în exemplul următor, Web cache-urile pot reduce semnificativ traficul pe legăturile de acces spre Internet ale instituțiilor. Prin reducerea traficului, instituția nu trebuie să mărească lărgimea de bandă aşa rapid, reducând astfel din costuri. Mai mult, Web cache-ul poate reduce traficul general din Internet, ceea ce duce la creşterea performanței pentru toate aplicațiile.

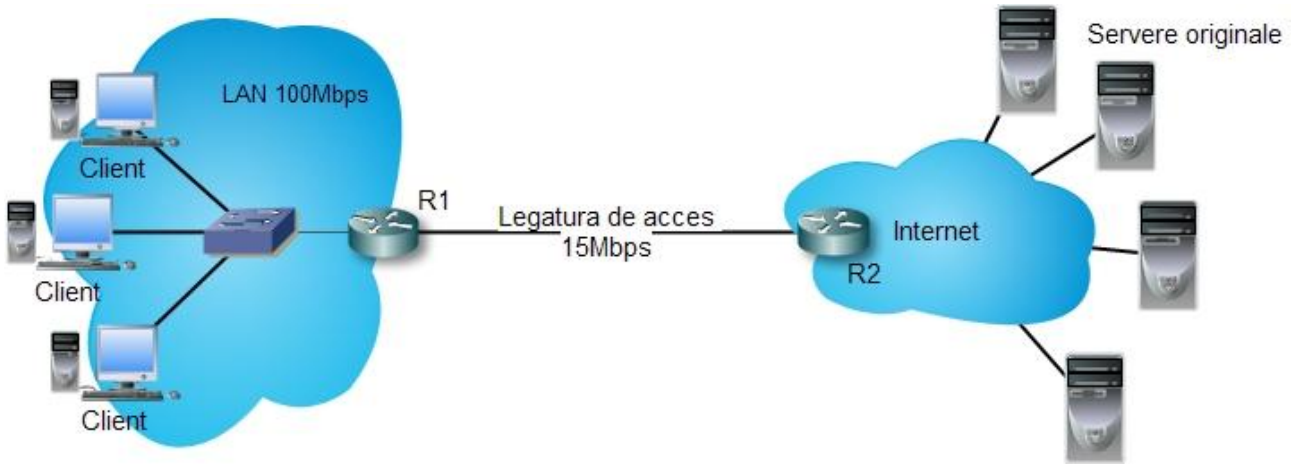


Figura 4.5 Gâtuire între rețeaua instituțională şi Internet

Pentru a studia mai îndeaproape beneficiile cache-urilor, să avem analizăm un exemplu în contextul din Figura 4.5. Figura prezintă două rețele – rețeaua instituțională şi restul Internetului public. Rețeaua instituțională este un LAN de mare viteză. Rețeaua instituțională este conectată la Internet print-o legătură de 15Mbps. Serverele Web ataşate la Internet sunt dispersate pe tot mapamondul. Să presupunem că dimensiunea medie a unui obiect este de 1Mb iar rata medie a cererilor provenite de la navigatoarele instituției către serverele originale este de 15 cereri pe secundă. Presupunem că cererile HTTP sunt mici şi nu creează trafic în rețele sau pe legătura de acces. Presupunem de asemenea că intervalul de timp dintre retrimiterea de către ruterul R2 a cereri HTTP(o datagramă IP) şi recepționarea răspunsului (de regulă mai multe datagrame IP)

este în medie 2 sec. Vom referi această întârziere (informal) sud denumirea de „întârzierea Internetului”.

Timpul total de răspuns – din momentul formulării cererii HTTP de către navigator şi până la recepționarea acelui obiect – este suma dintre întârzierea LAN, întârzierea de acces (între R1 şi R2) şi întârzierea Internet. În continuare vom efectua o estimare grosieră a acestei întârzieri. Intensitatea traficului în rețeaua LAN este :

(15 cereri/sec) * (1 Mb/cerere) / 100Mbps = 0,15

Pe când intensitatea traficului pe legătura de acces este:

15 cereri/sec) * (1 Mb/cerere) / 100Mbps = 1

O intensitate a traficului de 0,15 în rețeaua LA se traduce prin întârzieri de zeci de milisecunde; deci putem neglija această întârziere. Pe măsură ce intensitatea traficului se apropie de 1 (ca în cazul legăturii de acces din Figura 4.5), întârzierea de pe legătura de acces devine foarte mare şi poate creşte fără limită. Astfel, timpul mediu de răspuns pentru satisfacerea cererilor poate fi de ordinul minutelor, sau chiar mai mult, fapt care este inacceptabil pentru utilizatori. În mod clar trebuie făcut ceva.

O posibilă soluție este creşterea ratei de acces de la 15Mbps la, să zicem, 100Mbps. Aceasta va diminua intensitatea traficului pe legătura de acces la 0,15, rezultând întârzieri neglijabile între cele două rutere. În acest caz timpul total de răspuns va fi de circa 2 secunde (întârzierea Internet). Soluția implică modernizarea legăturii de acces de la 15Mbps la 100Mbps, cu implicații financiare serioase.

Acum să avem în vedere soluția alternativă a instalării unui Web cache în rețeaua instituțională. Soluția este ilustrată în Figura Figura 4.6. Ratele de succes (eng. hit rates) – fracțiunea de cereri satisfăcute de Web cache – variază în practică între 0,2 şi 0,7. Să considerăm că rata de succes are valoarea 0,4. Întrucât clienții şi Web cache-ul sunt conectați la acelaşi LAN de mare viteză, cererile vor fi satisfăcute aproape imediat de către cache, să zicem în 10milisecunde. Restul de 60% din cereri trebuie însă satisfăcute de serverele Web originale. Cu 60% din cereri traversând legătura de acces, intensitatea traficului se reduce de la 1 la 0,6. De regulă, o intensitate a traficului de 0,8 sau mai mică se traduce printr-o întârziere mică (zeci de msec), pe legătura de acces de 15Mbps. Această întârziere este neglijabilă în raport cu întârzierea Internet de 2 sec. În aceste condiții, întârzierea medie este:

0,4* (0,01 sec) + 0,6 * (2.01 sec)

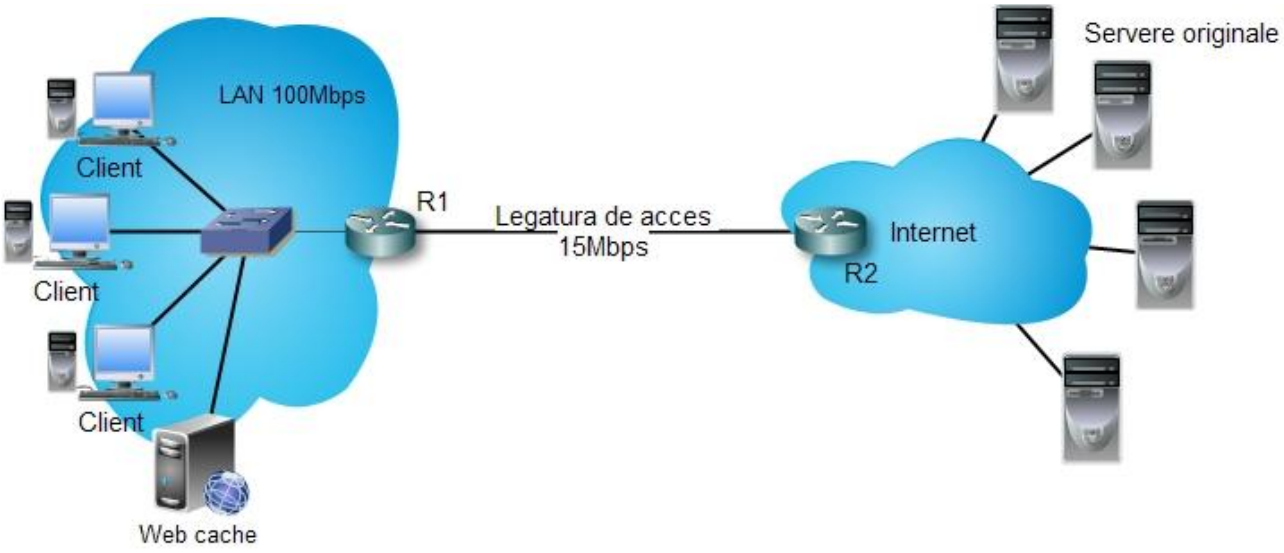


Figura 4.6 Adăugarea unui server proxy (Web cache) la rețeaua instituțională

Care este puțin mai mare de 1,2 secunde. Această soluție asigură chiar un timp de răspuns mai mic decât prima soluție, şi nu necesită creşterea lărgimii e bandă. Totuşi instituția trebuie să achiziționeze şi să instaleze un server proxy; costurile sunt însă reduse – multe web cache sunt bazate pe soft-uri libere (de ex. squid) ce pot rula pe PC-uri ieftine.

Prin utilizarea Reţelelor de Distribuţie a Conţinutului (eng. Content Distribution Networks, CDN), Web cache-urile joacă un rol din ce în ce mai important în Internet. O companie CDN instalează mai multe cache-uri distribuite în Internet, ceea ce duce la localizarea traficului. Există CDN-uri partajate (cum ar fi Akamai şi Limelight) şi CDN-uri dedicate (cum ar fi Google sau Microsoft).

Rezumat

HTTP este un protocol simplu de nivel aplicație. Clienții (navigatoarele Web) efectuează cereri folosind metoda GET (sau POST) iar serverul Web furnizează un răspuns, aceasta fiind o abordare clasică client-server. Pentru asigurarea unui transfer fiabil al mesajului de cerere de la client la server precum şi a mesajului de răspuns de la server la client HTTP utilizează TCP; acest lucru necesită stabilirea în prealabil a unei conexiuni TCP inițiată de către client. După schimbul TCP urmează transferul prospriu-zis al mesajului HTTP GET (sau POST) de la client către server şi primirea răspunsului. În cazul conexiunilor HTTP nepersistente, se stabileşte o nouă conexiune TCP de fiecare dată când clientul solicită un obiect de la server. În cazul conexiunilor HTTP persistente se pot transmite

mai multe mesaje HTTP de cerere urmate de răspunsurile corespunzătoare peste aceeaşi conexiune TCP, rezultând creşteri de performanță întrucât nu mai este necesară stabilirea de noi conexiuni TCP.

Caching-ul reprezintă salvarea locală a unei copii a unei informații (document Web şi nu numai) extrasă dintr-o locație aflată la distanță, astfel încât în momentul în care acea informație este solicitată din nou, poate fi extrasă din cache-ul local, fără a fi necesară extragerea informație in locația aflată la distanță.

Bibliografie

Andrew S. Tanenbaum, *Computer Networks*, 4/E, Prentice Hall, 2003

James F. Kurose and Keith W. Ross, *Computer Networking A Top Down Approach*, 5/E, Pearson Education, 2009

William Stallings, *Data and Computer Communications*, 9/E, Pearson Education, 2011