PROTOCOALE DE COMUNICAȚIE : Tema #3

Server Proxy pentru protocolul HTTP*

Responsabili: Dorinel FILIP, Radu-Ioan CIOBANU, Florin POP
Termen de predare: 15 MAI 2017

Cuprins

*Ideea acestei activiati practice are la baza o tema propusa la cursul COS-518: Advanced Computer Systems, tinut de Prof. Mike Freedman la Universitatea Princeton, USA.

Obiectivele temei

Scopul temei este realizarea unei aplicații de tipul Server Proxy pentru protocolul HTTP1.0, folosind *Berkeley sockets API*. Obiectivele temei sunt:

- parcurgerea și înțelegerea unui document de tip RFC;
- aprofundarea cunoștințelor privind protocolul HTTP;
- implementarea, folosind *Berkeley sockets API*, a unei aplicații conforme (pentru un scenariu restrâns) cu un RFC dat;
- cunoașterea modului de funcționare a unui proxy HTTP și implementarea unei variante minimale pentru acesta.

Enunţul Temei

Se dorește implementarea unui proxy HTTP minimal capabil să intermedieze comunicația între un browser web și un server HTTP, ambele folosind protocolul HTTP1.0.

Pentru numărul maxim de puncte este suficient să faceți implementarea unui server (single-threaded) folosind API-urile de multiplexare învățate în cadrul laboratorului. Implementarea unei variante mai eficiente, folosind thread-uri, va fi considerată bonus.

Pentru comunicarea pe rețea se va folosi exclusiv *Berkeley sockets API* (studiat în cadrul laboratorului).

Implementarea va consta într-un program C/C++ care primește ca parametru în linie de comandă un număr de port și ascultă pe toate adresele IP asociate stației, doar pe portul indicat, deservind ca server proxy HTTP pentru orice solicitare primește.

Modul de apelare a programului va fi:

```
./httpproxy <port_proxy>
```

Funcționarea unui proxy HTTP

La bază, protocolul HTTP este unul de tip client-server, în care entitatea client (de obicei un browser web) se conectează direct la entitatea server (un software specializat în livrarea de conținut web). Cu toate acestea, în anumite situații poate fi util să interpunem o nouă entitate, numită *proxy*, între server și client. Practic, acesta deservește ca intermediar între cele două entități (server și client). În cazul cel mai uzual, utilizarea proxy-ului este impusă din momentul lansării cererii (de către browser). Astfel, acesta, în loc să direcționeze cererile direct către server, le va trimite către proxy. Mai departe, proxy-ul deschide o conexiune cu serverul web și îi trimite acestuia solicitarea primită de la client. La primirea răspunsului, proxy-ul îl direcţionează mai departe către client.

Observăm că proxy-ul are atât rol de client (față de server-ul căruia îi solicită procesarea cererii), cât și de server (față de clientul inițiator al solicitării).

Printre motivele pentru care un astfel de server proxy ar putea fi util, enumerăm:

- **Performanța** - prin salvarea unor copii ale paginilor pe care le solicită, un proxy poate reduce numărul de cereri efectuate către exterior, în cazul în care o resursă este cerută în mod repetat.
- **Filtrarea conținutului** - deși în cazul cel mai simplu un server proxy doar preia o solicitare și o trimite mai departe, acesta poate fi folosit pentru impunerea anumitor politici privind accesul la diverse pagini web (de exemplu, refuzând cererile către conținutul nepermis utilizatorului dat).
- **Anonimizarea traficului** - în mod obișnuit, un server web jurnalizează toate toate cererile primite. Un astfel de jurnal conține cel puțin adresa IP a clientului care a făcut cererea, browserul folosit (numit User-Agent), momentul de timp și resursa solicitată. Dacă un utilizator nu dorește ca date ce permit

identificarea lui să ajungă într-un astfel de jurnal, poate folosi un proxy. Toate solicitările făcute printr-un proxy vor apărea, din perspectiva serverului, ca venind de la adresa IP a serverul proxy. Dacă un număr mare de utilizatori folosesc același proxy, atunci a face legătura între o stație și o solicitare anume devine foarte dificil.

Detalii de implementare

Specificații generale

Sarcina voastră este să scrieți programul pentru un proxy web capabil să aștepte cereri HTTP, să le redirecționeze către serverul responsabil de acea resursă, și să întoarcă răspunsul către client.

Tema poate fi realizată folosind C/C++. Pentru a fi punctată, aceasta trebuie să compileze fără erori și warning-uri în cadrul mașinii virtuale de aici, producând un fișier executabil numit `httpproxy`, care primește ca prim parametru în linie de comandă portul pe care va asculta.

NU folosiți valori hardcodate pentru port (ex. port 80).

Implementarea voastră nu trebuie să presupună că proxy-ul sau serverul au asignată o anumită adresă IP și nici că cererile vor origina întotdeauna de la o anumită adresă / același client.

Așteptarea cererilor

Atunci când este pornit, programul proxy trebuie să-și stabilească un socket TCP ce va fi folosit pentru primirea de noi cereri. Proxy-ul trebuie să asculte pe portul indicat ca parametru în linie de comandă și să aștepte noi cereri de conectare.

Atunci când apare o nouă cerere, proxy-ul o va citi și va verifica dacă aceasta conține o cerere HTTP validă. O cerere invalidă va avea ca răspuns un cod de eroare adecvat (vezi RFC 1945).

Parsarea URL-ului

Atunci când identifică o cerere validă, proxy-ul va trebui să parseze URL-ul din cadrul acesteia. El va avea nevoie de cel mult trei informații: host-ul căruia i se va solicita informația, port-ul serverului și calea către resursa solicitată.

Pentru detalii vezi man 7 url.

Verificarea cache-ului

Odată determinat obiectul web solicitat (ca fiind întregul URL), trebuie să verificați dacă acesta se află deja în cache-ul proxy-ului. Dacă da, atunci se va răspunde cu conținutul din memoria cache.

Pentru simplitate, nu trebuie să implementați algoritmul de invalidare a cache-ului din HTTP: cache-ul va fi curățat complet la pornirea proxy-ului, iar toate resursele vor fi păstrate în cache pentru un timp nedeterminat. De asemenea, nu este obligatoriu să trimiteți solicitări condiționale (ex. "If-Modified-Since") către server.

Dacă doriți, puteți - bineînțeles - să implementați suportul real pentru cache.

Preluarea datelor de la server

Dacă resursa nu a fost găsită în cache, odată parsat URL-ul, proxy-ul va deschide o conexiune către serverul remote (în cazul în care în URL nu apare niciun port, atunci se va folosi port-ul 80) și va trimite o solicitare pentru resursa cerută de client. Solicitarea trimisă va fi cea primită de la client.

Scrierea datelor în cache

După preluarea cu succes a resursei, aceasta va fi salvată de către proxy pe disc, astfel încât solicitările ulterioare să utilizeze copia locală, în loc să repete preluarea datelor de la server.

Nu trebuie să puneți în cache resursa dacă aceasta este marcată cu "no-cache" sau "private" (vezi RFC-ul). Tema trebuie să facă cache doar răspunsurilor cu cod 200 (OK). Nu trebuie să vă îngrijiți de alte solicitări care permit să fie cached (cum ar fi 410 - GONE).

Trimiterea datelor către client

Serverul trebuie să trimită răspunsul către client, prin intermediul socket-ului corespunzător. Odată tranzacția încheiată, proxy-ul va închide conexiunea cu clientul.

Testarea manuala a programului

Ca un test simplu, porniți programul folosind comanda din Enunţul Temei, apoi într-un terminal, lansați o cerere către acesta folosind telnet:

```
telnet localhost <PORT>
<port> Trying 127.0.0.1...  Connected to localhost.localdomain
(127.0.0.1).  Escape character is '^]'.
GET http://www.google.com/ HTTP/1.0
```

Dacă proxy-ul funcționează corect, atunci în consolă se vor afișa headerele, apoi pagina HTML a site-ului Google.com. Observați utilizarea URL-ului complet, în locul referinței relative (/). Programul vostru trebuie să permită ambele variante.

Pentru teste mai complexe, va trebui să folosiți un browser.

Configurarea unui browser pentru testarea temei

Observație privind testarea temei

În cazul în care nu veți face implementarea bonusului, întrucât programul vostru va rula pe un singur fir de execuție, în timp ce browserele sunt extrem de agresive în paralelizarea transferurilor, este posibil ca unele pagini să nu încarce toate resursele. Acest lucru este tolerat în cazul temei, atât timp cât programul funcționează corect cu pagini simple.

Configurarea Mozilla Firefox pentru a folosi proxy

Pentru a configura Mozilla Firefox să folosească un proxy, se pot urma pașii de mai jos:

1. Din meniul principal, se dă click pe pictograma *Options*;

2. Se navighează către Advanced → (tab-ul) Network și se dă click pe butonul "Settings" din aria "Connections";
3. În caseta de dialog se completează datele proxy-ului.

Configurarea Mozilla Firefox pentru a folosi HTTP1.0

În mod normal, Mozilla Firefox folosește versiunea 1.1 a protocolului HTTP atât pentru solicitările către servere, cât și către servicii de tip proxy. Pentru a configura Firefox să folosească HTTP1.0 în comunicația cu proxy-ul vostru, trebuie să urmați pașii de mai jos:

1. Navigați către "about:config";
2. Filtrați opțiunile după string-ul "network.http.proxy";
3. Identificați opțiunile "network.http.proxy.pipelining" și "network.http.proxy.version", apoi setați-le valorile la *false*, respectiv 1.0.

Bonus

Chiar și folosind API-ul de multiplexare a socketilor, implementarea are ca principal dezavantaj imposibilitatea de a procesa 2 sau mai multe cereri în paralel. Se oferă două puncte bonus pentru o implementare paralelizată a temei.

Pentru rezolvare, puteți folosi (la alegere) una din următoarele abordări:

- crearea de noi procese - *fork()*, *waitpid()*;
- crearea de noi fire de execuție - *pthread_create()*, *pthread_exit()*, etc.

Mai multe informații despre funcțiile Linux pentru paralelizarea programelor puteți găsi în manual:

- man 2 fork
- man pthread

Testarea temei

Soluția se va trimite sub forma unei arhive ZIP având numele conform sintaxei `<grupa>_<nume>_tema3.zip`. Arhiva trebuie să cuprindă:

- codul sursă aferent programului implementat;
- un fișier Makefile pentru generarea fișierului executabil proxy;
- un fișier README care să explice deciziile luate în implementare.

Pentru a fi notată, o temă trebuie:

- să compileze fără erori pe mașina virtuală oferită
- să poată fi folosită ca proxy pentru Mozilla Firefox
- să treacă testele publice postate aici. O temă care rezolvă cerința temei și trece testele publice va fi notată cu 6 puncte din 8.

Alte precizări privind notarea:

- Se oferă 2 puncte pentru folosirea corectă a socketilor, robustețea codului, verificarea situațiilor de eroare și conținutul fișierului README.
- Bonusul valorează 2 puncte și este oferit pentru trecerea simultană a testelor publice de aici și aici.
- Punctajul maxim pentru temă corespunde obținerii a 8 din cele 10 puncte aferente grilei de notare. Bonusul se acordă proporțional.