Universitatea POLITEHNICA din București
Facultatea de Automatică și Calculatoare

Structuri de Date 2019
Tema 2
Ierarhie DNS

deadline soft - 21 Aprilie 2019 23:55
deadline hard - 23 Aprilie 2019 23:55

Alexandra Pîrvulescu
Bogdan-Ștefan Neacșu

1 Obiective

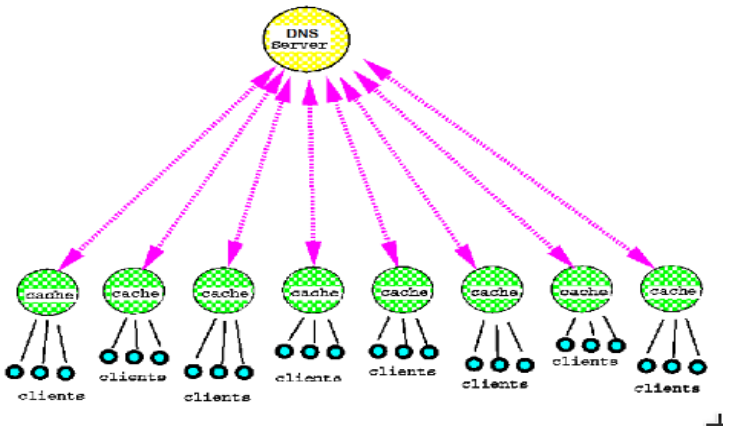
În urma realizării acestei teme, studentul va fi capabil:

1. să implementeze și să utilizeze arborii pentru rezolvarea unor probleme
2. să înțeleagă modul de funcționare a unei ierarhii DNS
3. să implementeze operații pe arbori

2 Descriere

Comunicarea în Internet se face pe baza adreselor IP. Deoarece ele sunt într-un format numeric, dificil de reținut pentru oameni, Internetul folosește rezolvarea DNS, care leagă denumiri ușor de reținut (e. g. acs.pub.ro) la adrese IP (e. g. 141.85.227.151). Serverele care fac aceste legături sunt organizate într-o structură arborescentă. În această temă veți implementa un model simplist de caching folosit într-o ierarhie DNS pentru aceste servere și câteva operații întâlnite în rezolvarea adreselor.

Conceptul de caching presupune ca adresele ce urmează a fi rezolvate să fie cât mai aproape de utilizatorul care le cere. Dacă la un moment dat, ele nu se află chiar în serverul DNS cel mai apropiat, vor fi salvate în memoria acestuia. La o eventuală cerere ulterioară, aceasta se va regăsi în cel mai apropiat server DNS. În imaginea de mai jos aveţi ilustrat un exemplu pentru o ierarhie pe două nivele.



Arborele va fi format din noduri ce reprezintă servere DNS. Fiecare server conține o listă de adrese pe care acesta poate să le rezolve.

Un nod are un nod părinte și 0 sau mai multe noduri copii.

Înainte de folosirea ierarhiei DNS, rețeaua va fi inițializată, proces prin care fiecare nod părinte al arborelui va fi actualizat cu adresele rezolvabile ale tuturor copiilor săi.

Prin rezolvarea adreselor ne referim la procesul prin care aflăm adresa IP a unui nume. O cerere DNS făcută către un server este rezolvată când în memoria serverului este găsită adresa cerută. Dacă adresa nu se află în memorie, ea este cerută de la server-ul părinte al serverului curent. Acest procedeu se repetă până când unul din servere reușește să rezolve adresa. Se garantează că serverul rădăcină (cel care nu are părinte) conține toate adresele pentru care se poate face cerere la un moment dat.

Pentru a face ierarhia rezistentă de defecțiuni, va trebui să ne asigurăm că, în cazul în care unul sau mai multe servere se defectează, ierarhia va fi în continuare funcțională. Pentru aceasta, ne vom asigura că funcțiile unui nod defectat vor fi preluate de nodul părinte al acestuia. Presupunem că nodul rădăcină nu se defectează niciodată.

3 Cerințe

3.1 3.1 Construcția arborelui (30 puncte)

Plecând de la fișierul *tree.in*, construiți arborele asociat ierarhiei DNS. În acest fișier veți găsi detalii legate de părintele fiecărui nod și lista de adrese rezolvabile.
Atenţie! Nodurile nu sunt date într-o ordine anume.
În fișierul *tree.out* veți afișa id-urile nodurilor din arbore, urmate de id-urile copiilor acestora. Mai multe detalii despre formatul celor două fișiere găsiți în secțiunea 4.

3.2 Inițializare ierarhie (30 puncte)

În urma inițializării trebuie să vă asigurați că orice nod părinte va avea în lista de adrese toate adresele pe care le au și copiii acestuia.
În fișierul *hierarchy.out* veți afișa id-urile nodurilor din arbore, urmate de adresele acestor noduri. Mai multe detalii despre formatul fișierului găsiți în secțiunea 4.

3.3 Cereri de la utilizatori (30 puncte)

În fișierul *users.in* aveți detalii despre id-ul server-ului DNS core-

spunzător fiecărui utilizator. În fișierul *queries.in* aveți cereri de adrese din partea utilizatorilor. Pentru fiecare cerere din *queries.in* va trebui să existe o linie corespunzătoare în fișierul *queries.out* în care vor fi afișate id-urile serverelor prin care s-a trecut, până la rezolvarea adresei cerute. Toate nodurile prin care s-a trecut pentru a rezolva adresa își vor actualiza lista de adrese cunoscute prin introducerea acestei adrese, dacă ea nu există deja. Mai multe detalii despre formatul celor două fișiere găsiți în secțiunea 4.

3.4 (Bonus) Toleranța la defecte (20 puncte)

La căderea unui server, trebuie asigurată în continuare ierarhia de DNS. Presupunem că serverul rădăcină (cel care nu are părinte) este singurul care nu se poate defecta. Atunci când un nod se defectează, părintele acestuia moștenește toți copiii săi și toți utilizatorii legați la copiii săi. Toate adresele stocate în nodul care s-a defectat vor fi pierdute.

4 Formatul fișierelor

- *tree.in*: Pe prima linie se găsește **n**, numărul de servere DNS. Pe următoarele **n** linii se găsește id-ul serverului curent, urmat de id-ul părintelui său, numărul de adrese rezolvabile de nodul curent, **m**, urmat de cele **m** adrese rezolvabile. Toate aceste date sunt separate prin spații.
  Observație: Serverul root are drept părinte id-ul -1, care nu corespunde niciunui server real.
- *tree.out*: Fișierul va conține **n** linii, pe fiecare linie aflându-se id-ul nodului curent urmat de id-urile copiilor săi, dacă există.
  Observație: Nici ordinea liniilor și nici ordinea copiilor pe linie nu contează.
- *hierarchy.out*: Fișierul va conține **n** linii, pe fiecare linie aflându-se id-ul nodului curent urmat de adresele rezolvabile de acesta.
- *users.in*: Pe prima linie se găsește **u**, numărul de utilizatori ai ierarhiei DNS. Pe următoarele **u** linii se găsesc perechi **id-user**, **id-server** separante prin spațiu. Utilizatorul cu **id-user** poate trimite cereri către serverul cu id-ul **id-server**.
- *queries.in*: Pe prima linie se găsește **c**, numărul de comenzi. Pe următoarele **c** linii se găsește una dintre cele două tipuri de comenzi:

  - comanda **q**: linia este de forma "**q id-user address**". Utilizatorul cu id-ul **id-user** trimite o cerere DNS către serverul pe care îl are configurat (conform fișierului *users.in*) pentru rezolvarea adresei **address**.
  - comanda **f**: linia este de forma "**f id-server**". Aceste comenzi vor apărea doar în testele pentru bonus și reprezintă căderea serverului cu id-ul **id-server**.

- *queries.out*: Pentru fiecare comandă de tip **q** din fișierul *queries.in* va exista o linie corespunzătoare de forma "id-server-1 id-server-2 ... id-server-n" reprezentând id-urile server-elor prin care a trecut cererea până la rezolvarea adresei.

5 Exemplu

tree.in:
6
0 -1 5 addr1 addr2 addr3 addr4 addr6
1 0 1 addr1
2 0 1 addr2
3 0 1 addr3
4 1 1 addr4
5 1 2 addr5 addr6

Avem 6 noduri, fiecare nod reprezentând un server. Primul nod are id-ul 0 şi este nodul rădăcină, deoarece are -1 trecut la id-ul părintelui. El are 5 adrese în memorie (şirurile de caractere "addr1", "addr2" etc.). Al doilea nod are id-ul 1, nodul părinte este nodul cu id-ul 0 şi o adresă în memorie ("addr1"). Ş.a.m.d.

tree.out:
0 1 2 3
1 4 5
2
3
4
5

Nodul 0 are copiii 1, 2, 3.
Nodul 1 are copiii 4 şi 5.
Nodurile 2, 3, 4 şi 5 nu au niciun copil (sunt frunze).

hierarchy.out:

0 addr1 addr2 addr3 addr4 addr5 addr6
1 addr1 addr4 addr5 addr6
2 addr2
3 addr3
4 addr 4
5 addr5 addr6

După iniţializarea ierarhiei (cerinţa 2), nodul 0 are toate adresele în memorie, nodul 1 are adresele addr1 addr4 addr5 addr6, nodul 2, addr2, nodul 3, addr3, nodul 4, addr4, iar nodul 5, addr5 şi addr6.

users.in:
2
0 2
1 4

Avem doi utilizatori legaţi la reţea.
Utilizatorul 0 este legat la nodul cu id-ul 2, iar utilizatorul 1 este legat la nodul cu id-ul 4.

queries.in:
8
q 0 addr2
q 0 addr1
q 0 addr1
q 1 addr4
q 1 addr1
q 1 addr2
f 4
q 1 addr4

Avem 7 cereri. Ultimele două cereri pot apărea doar în testele pentru bonus.
Utilizatorul 0 cere adresa 2, utilizatorul 0 cere adresa 1 etc.
f 4 reprezintă căderea nodului cu id-ul 4.

queries.out:
2
2 0
2
4
4 1
4 1 0
1

Utilizatorul 0, legat la serverul 2, cere adresa addr2, care se află în memoria serverului 2. De aceea se va afişa pe prima linie 2.
Utilizatorul 0 cere apoi adresa addr1. Cum aceasta nu se află în nodul 2, ci în părintele acestuia, care are id-ul 0, se va afişa 2 0.
Utilizatorul 0 cere apoi din nou adresa addr1. Aceasta se va afla de această dată în nodul cu id-ul 2 (s-a făcut caching pentru această adresă). De aceea se va afişa pe linia a treia indicele 2.
În mod similar se afişează şi pentru următoarele trei cereri.
La ultima cerere (cea după defectarea nodului 4), utilizatorul cu id-ul 1 va fi acum legat la nodul cu id-ul 1. Când acesta lansează apoi o cere pentru adresa addr4, ea va fi rezolvată de primul nod, nodul 1, care este şi afişat.

6 Restricții și precizări

- Temele trebuie să fie ıncărcate pe vmchecker. NU se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.
- O rezolvare constă ıntr-o arhivă de tip zip, care cont , ine toate fișierele sursă, alături de un Makefile, ce va fi folosit pentru compilare, și un fișier README, ın care se vor preciza detaliile de implementare. Toate fișierele se vor afla ın rădăcina arhivei.
- Punctajul alocat pentru Readme este 5p iar pentru coding style este tot 5p.
- Makefile-ul trebuie să aibă obligatoriu regulile pentru build și clean. Regula build trebuie să aibă ca efect compilarea surselor și crearea binarului.
- Deadline-ul soft al temei este 21 aprilie. Se vor accepta trimiteri de teme până pe 23 aprilie, cu depunctare de 10 puncte pe zi.