ASI 2021 – MSR1 – Tema 2

Cerinţă

Se vor crea două programe în Python 3.x – un client care va rula în mai multe instanțe și un server unic - care să permită rezolvarea distribuită în rețea a unei probleme de tip computațional intensivă, similară cu „*mining*”-ul Bitcoin (dar simplificată).

Descriere

Serverul va asculta pe cel puţin un port, şi anume port= (nr. echipei+12)*100 (pentru a fi >1024). Se pot folosi porturi suplimentare după dorinţă.

Echipele cu număr par vor folosi protocolul TCP iar celelalte UDP – vezi funcţiile corespunzătoare în lab de sockets. Cei care folosesc UDP nu trebuie să implementeze metode suplimentare de retransmisie (pentru a nu le complica sarcina față de cei cu TCP).

Programul server va avea ca parametri în linia de comandă un număr *n* și un șir de caractere *string* (reprezentarea text a unui număr hexazecimal de genul A7C02215FC). Alternativ, dacă se apelează fără parametri, el va cere introducerea lui *n* și *string*.
Scopul va fi găsirea de către client a unei valori hexazecimale numite *seed* care, concatenată la începutul lui *string* exprimat în hexazecimal, să dea ca rezultat un *hash* de tip MD5 al numărului hexazecimal concatenat *seed* | *string* care să înceapă cu *n* octeți de valoare 00. Simbolul | este operația de concatenare, de exemplu 1234 | ABCD = 1234ABCD

De exemplu, pentru *n=3*, un *hash* MD5 acceptabil va fi de forma:

0x*000000*51893FD5680A631B2419C1445E05

Lungimile lui *seed* și *string* sînt la alegerea voastră. Lungimea MD5 este standard (16 octeți).
Întrucît nu există o formulă matematică pentru calculul unor numere care să producă un *hash* MD5 impus, rezultă că problema se rezolvă prin încercări. Timpul de calcul crește exponential cu mărimea lui *n*. Acesta este principiul cîștigării de *bitcoins* prin “minare”, întrucît efortul computational este foarte ridicat și concurența este mare (doar primul care găsește este răsplătit).

Programul client va avea ca parametru în linia de comandă adresa serverului și se vor porni mai multe instanțe ale acestuia (minim două), de la orice adresă din rețea. Fiecare client, la momentul conectării la server, se va înregistra la server, primind un identificator (1, 2, ...), serverul avînd în fiecare moment evidența și adresele tuturor clienților conectați.
Serverul va trimite fiecărui client conectat datele problemei (*n* și *string*).

Clientul va folosi funcția *md5.digest()* sau *md5.hexdigest()* [1] pentru a calcula MD5-ul (*hash*-ul) de 16 octeți. Primul client care va găsi o soluție a problemei o va transmite la server, și acesta va afișa soluția și identificatorul clientului cîștigător. De asemenea, va transmite tuturor celorlalți clienți să înceteze lucrul.

În timpul funcționării, fiecare client va afișa și va transmite periodic către server numărul de hash-uri efectuate pînă atunci, cu o anumită periodicitate la alegere (de exemplu un mesaj la fiecare 10.000 hash-uri încercate) a.î. activitatea să fie vizibilă. Serverul va afișa cîte o linie pentru fiecare client, pe care se vor raporta adresa, identificatorul și nr. de hash-uri efectuate pînă acum (în timp real), plus eventuala stare de cîștigător.

Fiecare echipă trebuie să producă o pereche client-server compatibile între ele (este recomandabil ca cei 2 membri ai echipelor să-și împartă între ei lucrul, unul făcînd clientul, celălalt serverul, după ce au stabilit foarte clar formatul pachetelor schimbate între acestea); întrucît sînt diferenţe la numărul de port, dar vor

exista şi alte diferenţe datorită faptului că nu este vorba de un protocol standard (este o variantă simplificată față de bitcoin-ul real), nu se cere compatibilitate între clienţii şi serverele de la echipe diferite. Mai mult, voi verifica ca programele să nu fie identice între echipe!

Orice detaliu necesar funcționării și care nu a fost specificat mai sus este la alegerea voastră, şi-l veți comenta în fişierul sursă şi/sau în *readme.txt*.

Mod de predare

În directorul personal al echipei curente (de pe *matrix*) se va găsi un director *tema2* (fără spații sau litere mari) care va conţine 3 fişiere:
- programul *client.py* (executabil)
- programul *server.py* (executabil)
- un fişier *readme.txt* care va descrie pe scurt modul de operare a programelor , și va documenta modul în care se transmit diferitele mesaje schimbate între client şi server prin rețea (stringul original, numărul n, numărul de hash-uri efectuate, faptul că s-a găsit soluția și că ceilalti clienți trebuie să se oprească, etc). Practic, aceste mesaje formează „protocolul aplicației", care nefiind standard, trebuie documentat (vezi ca exemplu mesajele de protocol DNS sau FTP, TFTP etc studiate la curs; fiecare tip de mesaj/pachet are un anumit format, un anumit număr de octeți – care poate fi fix sau variabil, în ultimul caz trebuind stabilită o modalitate de determinare la recepție a lungimii mesajului, etc). În acest caz, fiind puține tipuri de mesaje, se va alege o structură cît mai simplă – de exemplu, ceva asemănător cu TFTP-ul și nu cu FTP-ul care are mesaje complexe.

Mod de testare - notare

Pentru fiecare echipă:
- voi deschide 3 terminale (1 server, 2 clienţi), voi intra în directorul personal al echipei
- voi citi instrucţiunile din *readme.txt*
- voi verifica funcționarea cf. specificațiilor (pentru un *n* rezonabil de mic)
- voi examina sumar sursele, urmărind cel puţin:
  - să se fi respectat folosirea protocolului UDP/TCP şi a numărului de port
  - să nu fie surse identice pentru echipe diferite

(nerespectarea celor două cerinţe de mai sus va duce la obţinerea notei 1, respectiv 0)

Punctaj: 25% existenţa fişierelor şi calitatea documentaţiei (inclusiv, şi în special, documentarea mesajelor de protocol), 75% funcţionarea.

Bibliografie și resurse:

[1] *https://docs.python.org/3/library/md5.html*