

Cerință

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 (mult simplificat).

Descriere

Serverul va asculta pe cel puțin un port, și anume $\text{port} = (\text{nr. echipei} + 10) * 100$ (pentru a fi > 1024). Se pot folosi porturi suplimentare după dorință.

Echipele cu număr par vor folosi protocolul UDP iar cele cu număr impar, TCP. 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 gășirea 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 SHA-256 al numărului hexazecimal concatenat $\text{seed} \mid \text{string}$ care să înceapă cu n cifre 0. Simbolul \mid este operația de concatenare, de exemplu $1234 \mid ABCD = 1234ABCD$

De exemplu, pentru $n=6$, un *hash* SHA-256 acceptabil ar fi de forma:

0x000007d41651593c5fe5c006fa5752b37fdff7bc4e843aa6af0c950f4b9406

Lungimile lui *seed* și *string* sînt la alegerea voastră. Lungimea SHA-256 este standard (256 biți, 64 cifre hexazecimale, 32 octeți).

Întrucît nu există o formulă matematică pentru calculul unor numere care să producă un *hash* impus (funcția *hash* nu este inversabilă), rezultă că problema se rezolvă prin încercări. Timpul de calcul crește exponențial cu mărimea lui n . Acesta este principiul cîștigării de *bitcoins* prin “minare”, întrucît efortul computațional 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 pentru teste), 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 metoda *hexdigest()* de aici: [1] pentru a calcula *hash*-ul SHA-256. 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 diferențe la numărul de port, dar vor

exista și alte diferențe datorită faptului că nu este vorba de un protocol standard (este o variantă mult simplificată față de bitcoin-ul real), nu se cere compatibilitate între clienții și 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, și-l veți comenta în fișierul sursă și/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 conține 3 fișiere:

- programul *client.py* (executabil)
- programul *server.py* (executabil)
- un fișier *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 și server prin rețea: stringul original, numărul n , numărul de hash-uri efectuate, faptul că s-a găsit soluția și că ceilalți 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 prezentate 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 cât 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 clienți), voi intra în directorul personal al echipei
- voi citi instrucțiunile din *readme.txt*
- voi verifica funcționarea cf. specificațiilor (pentru un n rezonabil de mic)
- voi examina sumar sursele, urmărind cel puțin:
 - să se fi respectat folosirea protocolului UDP/TCP și a numărului de port
 - să nu fie surse identice pentru echipe diferite

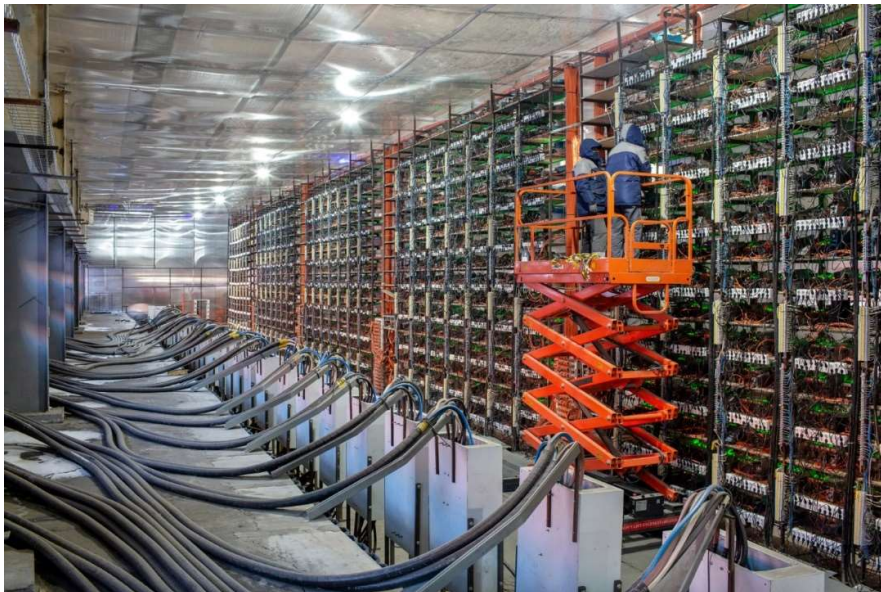
(nerespectarea celor două cerințe de mai sus va duce la obținerea notei 1, respectiv 0)

Punctaj: 25% existența fișierelor și calitatea documentației (inclusiv, și în special, documentarea mesajelor de protocol), 75% funcționarea.

Fapt divers: în minarea reală a Bitcoin, numărul de zerouri este mare: pt. mai 2024, 19 zerouri, și trebuie calculate în medie 88 T (88 milioane de milioane) de *hash*-uri (se numește *bitcoin difficulty* numărul mediu de *hash*-uri care trebuie calculate pentru a găsi soluția; acest parametru se ajustează de către rețea periodic, la fiecare 2016 blocuri, pentru a asigura un timp mediu de găsim a soluției constant, de cca. 10 minute, ceea ce înseamnă că un bloc nou este creat de cineva la fiecare 10 minute. Așadar, ajustarea se face la cca. 2016 blocuri * 10 min/bloc = 2 săpt. Ajustarea se face în funcție de numărul total de mineri la un moment dat și de puterea echipamentelor folosite de aceștia. Deci dintre toți minerii din lume care lucrează la un moment dat, cineva găsește o soluție după aprox. 10 minute.

Puteți urmări parametrii curenți ai rețelei (dificultatea, blocurile câștigătoare, numărul de *hash/s*) aici [2][3][4]

De asemenea, 18 aprilie 2024 a fost o dată importantă, căci a avut loc cel mai recent *Bitcoin Halving*, care se întâmplă odată la cca. 4 ani, recompensa pentru un bloc a scăzut de la 6.25BTC la 3.125BTC (aprox. 196000 USD la cursul zilei). Această recompensă se împarte între toți minerii care au participat la găsim a *hash*-ului câștigător, așa-zisul *mining pool*. În plus, minerii primesc BTC și în urma *transaction fees*.



Exemplu de „fermă” de minare, 2024.

Bibliografie și resurse:

- [1] <https://docs.python.org/3/library/hashlib.html>
- [2] <https://btc.com/stats/diff>
- [3] <https://whatsonchain.com/blocks>
- [4] <https://www.blockchain.com/explorer/blocks/btc>