Infokommunikációs hálózatok fizikai modelljei

Rudolf Ádám, Fizikus M.Sc., Számítógépes fizika szakirány

2015. május

1. feladat

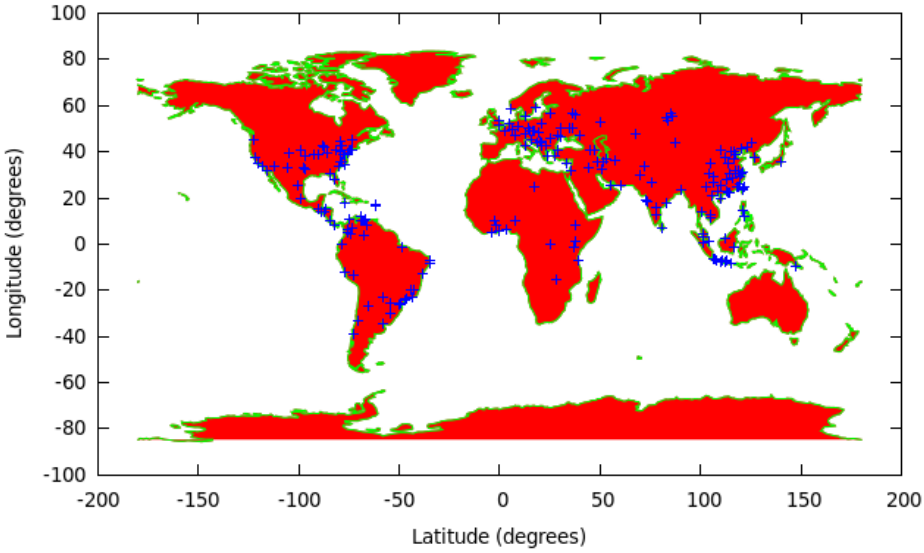
A feladatban több, különböző földrajzi helyen lévő számítógép válaszidejét kellett megmérni a Linux `ping` programjának segítségével. Az IP címek listáját a http://www.proxyserver4free.com/ oldalról vettem, ahol várossal és IP címmel együtt fel van sorolva 1 000, többnyire HTTP proxy szerver. A feladatban ezeket használtam a méréshez.

A folyamatok automatizálásához minden esetben Python scripteket írtam.

A http://pythonhosted.org/python-geoip/ oldalon találtam a Python `geoip` könyvtárát, ami tartalmaz egy `lookup` nevű metódust egy IP cím geokoordinátáinak meghatározására. Szúrópróbaszerűen ellenőrizve azt találtam, hogy a fent említett listában felsorolt címek megegyeznek a Python függvény által megadott hellyel. Megjegyzendő, hogy tapasztalataim szerint a metódus csak város pontosság erejéig adja meg a koordinátákat (vagyis egy városban lévő szerverekhez ugyanazt a koordinátát adja meg), de úgy ítéltem meg, hogy globális méretekben az ebből eredő hiba elhanyagolható. Annak ellenére, hogy saját koordinátáimat pontosabban meg tudom határozni, a következetesség kedvéért ezek helyett is a Python függvény által megadott koordinátákat fogom használni.

A https://www.whatismyip.com/ oldal segítségével meghatároztam a saját IP címemet. Ellenőrizni is lehetett, hogy valóban a saját helyünket adja-e meg. Ez alapján a számítógépem kívülről látható IP címe 80.99.69.6.

A geoip `lookup` függvénye a 47,5; 19,0833 koordinátákat adta meg a számítógépemhez. A következő számításokban ezt használtam, mint saját koordináta.



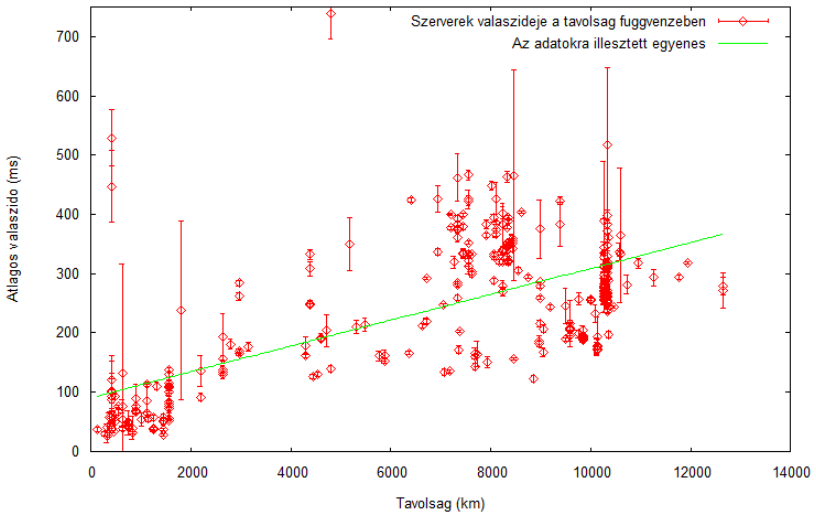
1. ábra. A http://www.proxyserver4free.com/ oldalon talált szerverek földrajzi eloszlása. A világ majdnem minden tájáról vannak címek.

A távolságot a haversine formulával fogom számolni:

$$d = 2 \cdot R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (1)$$

ahol φ_1 és φ_2 az egyik, illetve másik pont földrajzi szélességét, λ_1 és λ_2 pedig hosszúsági koordinátáit jelentik. R a Föld sugara, amit 6371 km-nek vettem.

A fent említett listából kigyűjtöttem a tiszta IP címeket, file-ba mentettem, és Python programot írtam, ami mindegyikre meghívja a `ping` függvényt, leméri a válaszidejét, meghatározza a földrajzi koordinátáit, és leméri a távolságot a saját koordinátáimtól. A pinget 10-szer küldtem el, és a válaszidőket átlagoltam. Az sikeres eredményeket (ha legalább 1 próbálkozásra érkezik válasz, azaz a veszteség nem 100%, és meghatározható a földrajzi helyzete) kimenti egy szöveges file-ba. A `ping` program standard devianciát is számol, ezt is elmentettem, mint a mérés hibája. Mivel a program hosszú ideig futott, és többször is összeomlott, az 1 000-ből 535 címre futtattam le, amiből 270 válaszolt.



2. ábra. Az áltagos válaszidők a földrajzi távolság függvényében, az illesztett egyenessel.

A földrajzi távolság függvényében az átlagos válaszidőket az 2. ábra tartalmazza az adatokra illesztett egyenessel együtt. Ha a földrajzi távolságot s-sel jelöljük, a körüljárási időt pedig t-vel, akkor a kapott $t = a \cdot s + b$ egyenes paraméterei:

$$a = (0,0218 \pm 0,0014)\text{ms/km}$$

$$b = (90 \pm 11)\text{ms}$$

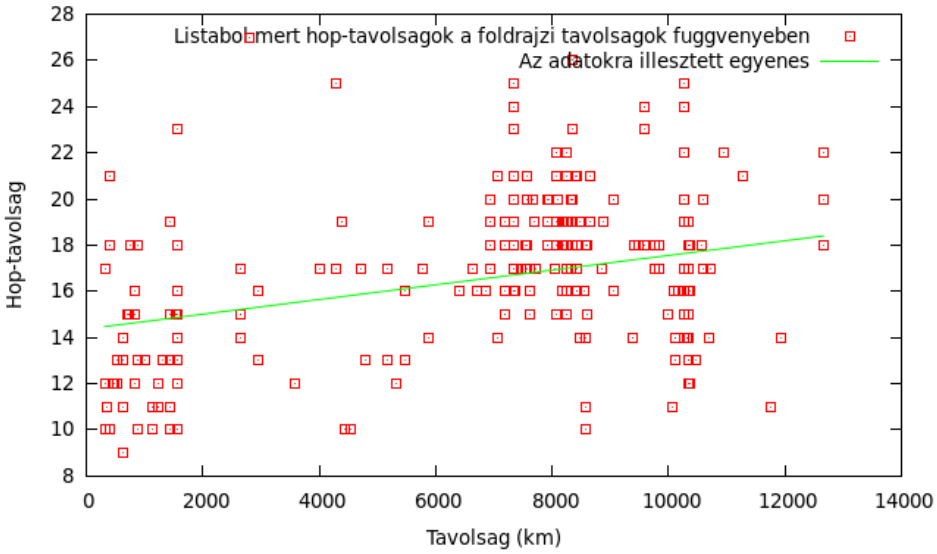
Ez alapján durván azt mondhatjuk, hogy a csomagok terjedési sebessége az interneten $1/a = (46 \pm 2,9)\text{km/ms}$.

2. feladat

A második feladat a hop-távolság vizsgálata volt a `traceroute` program segítségével. Először az előző feladat címlistáján, majd véletlenszerűen generált IP cím listán kellett vizsgálni a hop-távolság, és a földrajzi távolság összefüggését, azaz hogy hogyan függ az, hogy egy adott számítógépet hány kiszolgálón keresztül érünk el attól, hogy milyen messze van földrajzilag. Azután a hop-távolságok eloszlását kellett megvizsgálni.

2.1. Valós címlista

Python programot írtam, ami meghívja a `traceroute` Linux programot, és leméri, hogy hány ugráson keresztül jutottunk el a kívánt címig. Mivel gyakori, hogy a routerek blokkolják az ICMP csomagokat - amiket a `traceroute` alapértelmezett beállításban használ -, a programot a `-T` kapcsolóval hívtam meg, ami TCP próbacsomagokat küld ki. Hogy még nagyobb eséllyel kapjak választ, a várakozási időt az alapértelmezett 5 mp helyett 15 mp-re állítottam. A maximális TTL 30 volt, vagyis ha egy címet 30-nál több lépésben érnénk el, a cím nem került bele a listába. Mivel ez a mérés időigényesnek bizonyult, nem 1 000 címre futtattam a programot, csak 733-ra, amik közül 306 adott feldolgozható választ. A kapott hop-távolságokat a földrajzi távolság függvényében a 3. ábra tartalmazza.

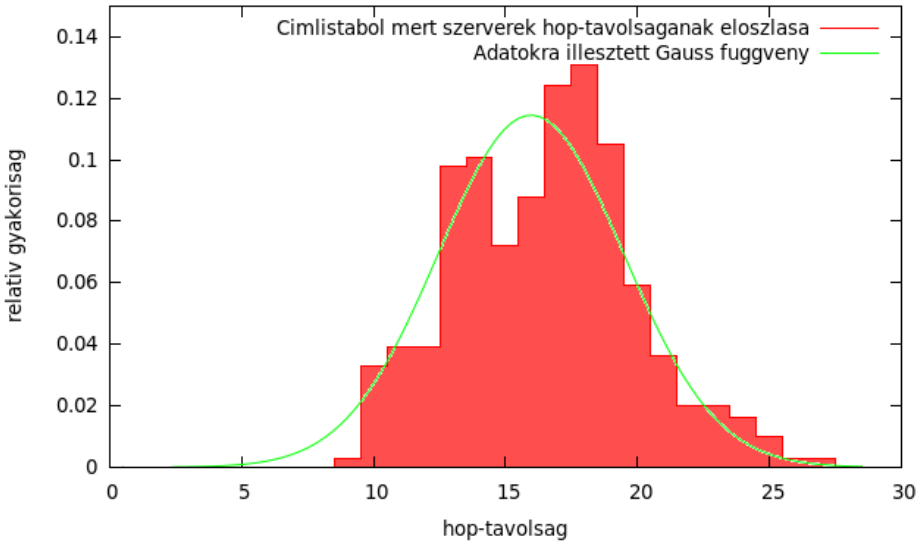


3. ábra. A listabeli címekre kapott hop-távolságok a földrajzi távolságok függvényében az adatokra illesztett egyenessel.

A hop-távolságokra egyenest is illesztettem a trend szemléltetése kedvéért, de ennek kicsi a meredeksége, és nagy a hibája (kb. 20%), így nem tudok egyértelmű lineáris kapcsolatot megállapítani a távolság, és a hop-távolság között. Az átlagos hop-távolság 16,5.

A hop-távolságokból hisztogramot készítettem, amit lenormáltam, és Gauss-függvényt illesztettem rá. Ez a 4. ábrán látható. A függvény jól illeszkedik, a közepe $16,0 \pm 0,23$-nál van, a szélessége pedig $3,5 \pm 0,23$.

Mindezekből arra következtetek, hogy nincs egyértelmű, közvetlen összefüggés a földrajzi- és hop-távolság között, vagy más hatások, amik valószínűleg az Internet topológiájából erednek, erősebbek.



4. ábra. Listából mért címek hop-távolságának eloszlása az adatokra illesztett Gauss-függvénnyel.

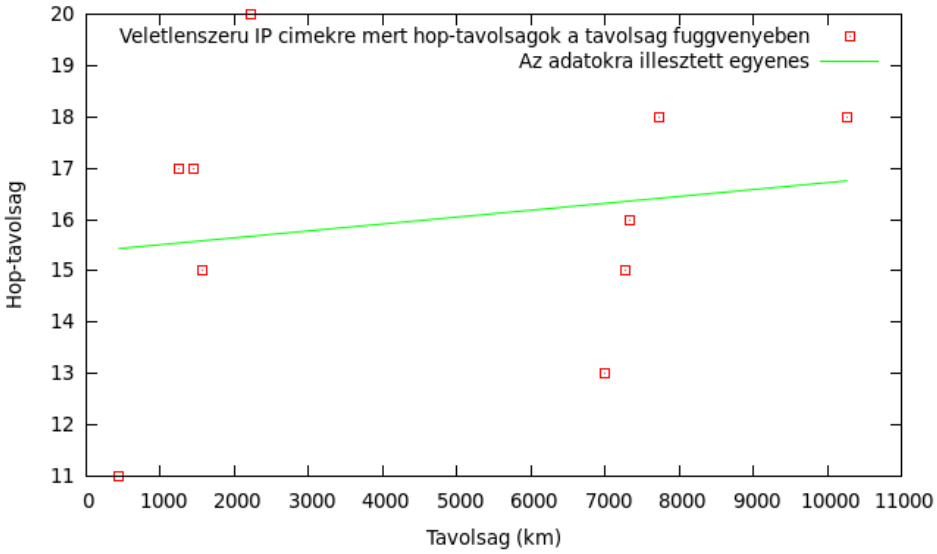
2.2. Véletlenszerűen generált címek

A feladat második részében ugyanezt a vizsgálatot kellett végrehajtani véletlenszerűen generált IP címekkel. Természetesen megkötés volt, hogy a címek publikusak legyenek. A privát címeket az `RFC 1918` szabvány definiálja. Eszerint 3 tartományba tartoznak a privát címek:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255

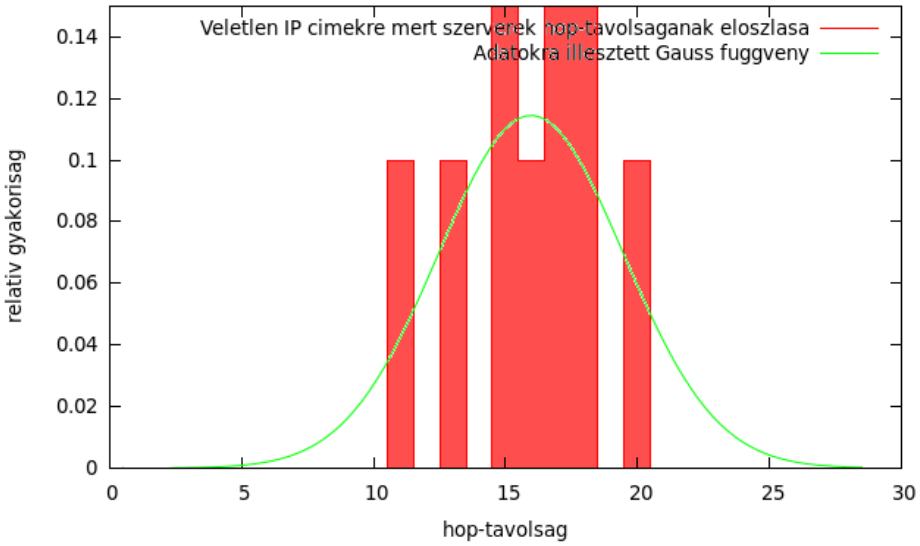
Függvényt írtam, ami a 0.0.0.0 - 255.255.255.255 tartományban véletlen címeket generál, ami ha a privát tartományba esik, újat generál, amíg publikusat nem kap.

Ilyen módon listát generáltam, amikre ugyanúgy lefuttattam a fent említett programot. Ezen címek nagyon kis része válaszolt (kb. 0,048%), így belátható időn belül csak nagyon kevés cím válaszolt (10 db). Az listából vett címekhez hasonlóan a távolság - hop-távolság összefüggését a 5. ábra, a hop-távolságok eloszlását pedig a 6. ábra tartalmazza.



5. ábra. A véletlenszerűen generált címekre kapott hop-távolságok a földrajzi távolságok függvényében az adatokra illesztett egyenessel.

Az elsőre illesztet egyenes meredeksége itt is kicsi, és hibája közel 200%. Az eloszlásra illesztett Gauss-függvény közepe $16,1 \pm 0,37$, szélessége pedig $2,1 \pm 0,37$. Az átlagos hop-távolság pontosan 16. Vagyis a listából vett, és a véletlenszerűen generált címek statisztikája gyakorlatilag megegyezik.



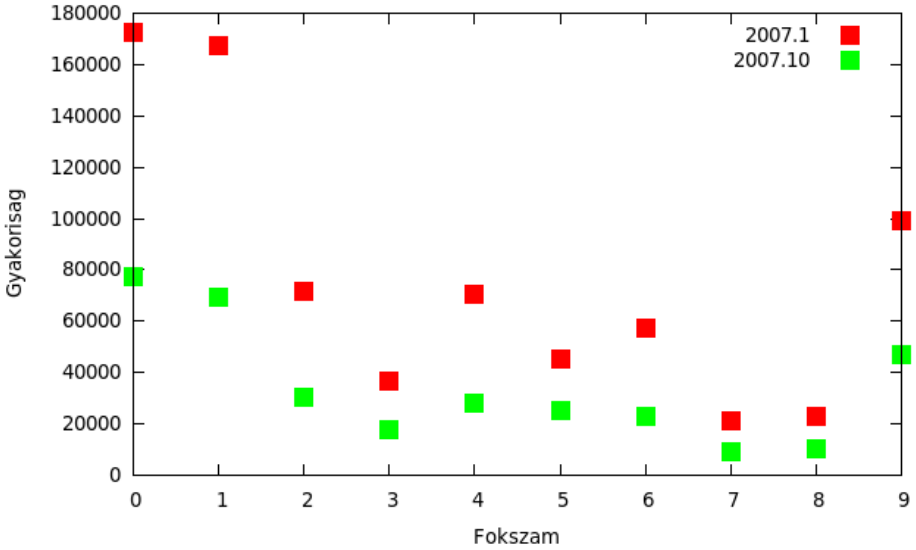
6. ábra. Véletlenszerűen generált címek hop-távolságának eloszlása az adatokra illesztett Gauss-függvénnyel.

3. feladat

Ebben a feladatban a www.netdimes.org oldalról letöltött éllista alapján kellett az internet topológiáját vizsgálni 2 külön időpontban. Ez a két időpont a 2007 januári, és októberi adatsor volt.

Python programot írtam, ami a letölthető CSV állományok alapján összeszámolja, hogy egy-egy egyedi cím hányszor fordul elő a listában, vagyis megméri a fokszámát, majd hisztogramot készít a fokszámokból. Megjegyzendő, hogy sok cím "?"-et tartalmazott, ezeket nem tudtam egyértelműen sehova sem sorolni, így egyszerűen kihagytam őket.

A két adatsorról így készített fokszám eloszlást a 7. ábra tartalmazza. Meglepő, hogy a számítás alapján egyik pontnak sem volt 9-nél több szomszédja, és az eloszlás lineáris skálán sem tűnik gyorsan levágónak.



7. ábra. A 2007 januárban, és októberben mért internet topológia alapján számolt fokszám eloszlások. Eredeti adatok forrása: www.netdimes.org.