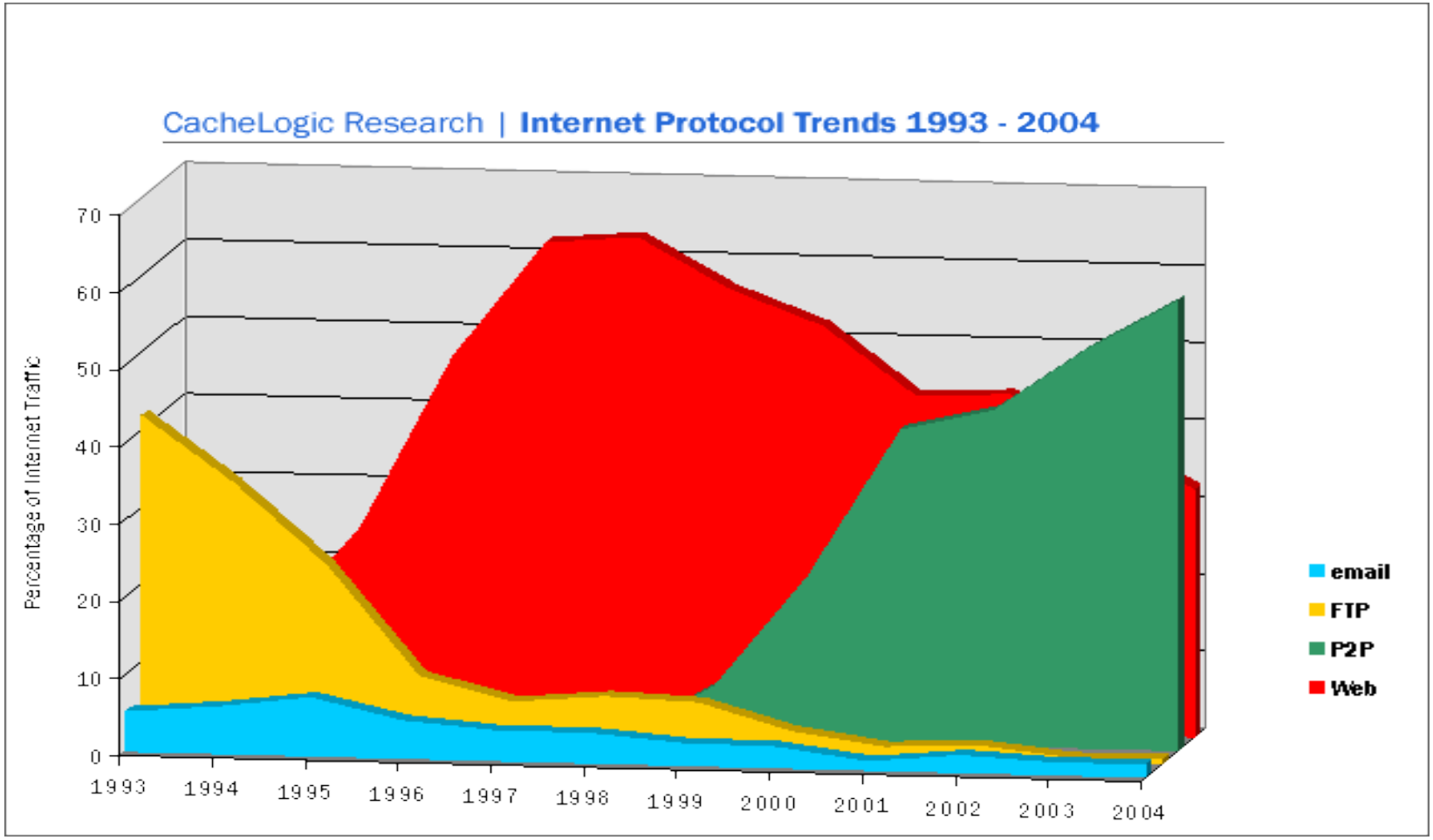
Számítógép hálózatok és osztott rendszerek

Peer-To-Peer Hálózatok

Forgalom az Interneten



http://www.cachelogic.com/research/2005_slide07.php#

Definíció

„Egy Peer-to-Peer (P2P) hálózat egy kommunikációs hálózat számítógépek között, melyben minden résztvevő mind kliens, mind szerver feladatokat végrehajt.“

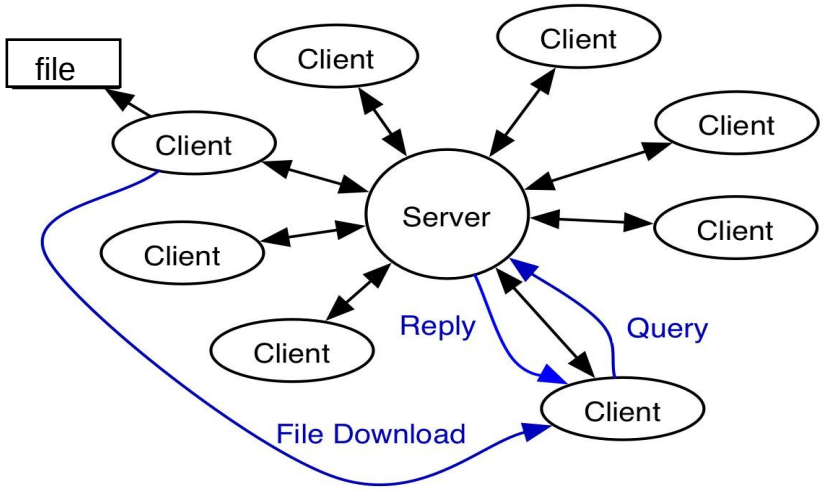
- Mi nem P2P hálózat?
  - nem Client-Server hálózat!

Napster története

- Shawn (Napster) Fanning
  - 1999 június megjelentette a Napster beta verzióját
  - Cél: File-sharing rendszer
  - Valóságban: Zene-cserebörze
  - 1999 ősszel Napster lett „Download of the Year“
- 2000 júniusában a zene ipar szerzői jogi pere
- 2000 végén együttműködési szerződés Fanning és Bertelsmann Ecommerce között
- Azóta Napster egy kommerciális Fájlmegosztó

Hogy működik Napster?

- Kliens-Szerver struktúra
- Szerver tárol
  - Indexet meta-adatokkal
    - fájlnév, dátum, stb.
  - A résztvevő kliensek kapcsolatainak táblázatát
  - A résztvevő kliensek fájljainak a táblázatát
- Lekérdezés (query)
  - Kliens kérdezi a fájl nevet
  - Szerver kikeresi a megfelelő résztvevőket
  - Szerver válaszol, kinek van meg a fájl (tulajdonos kliensek)
  - Kérdező-Kliens letölti a fájlt a tulajdonos-klienstől

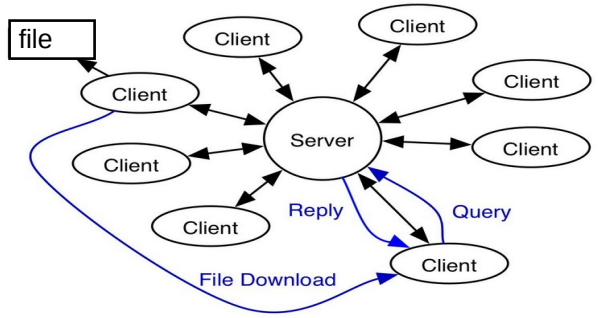


Mennyire jó Napster?

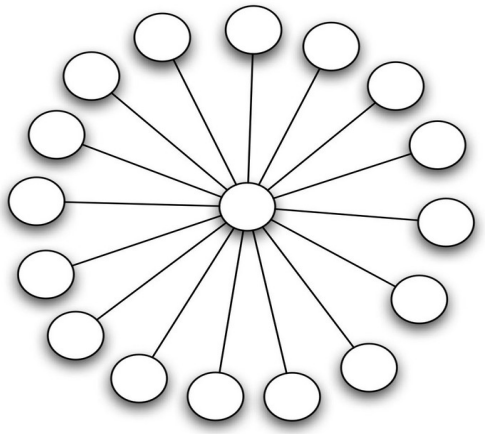
- Előnyök
  - Egyszerű
  - A fájlokat gyorsan és hatákonyan megtalálja
- Hátrányok
  - Centralizált struktúra megkönnyíti a cenzúrát, ellenséges beavatkozásokat és kieséseket
  - Nem skáláz
    - növekvő résztvevő számmal romlik a kiszolgálás minősége
    - a tár a szerveren véges
- Tanulság
  - Napster nem kielégítő P2P megoldás
  - A letöltéstől eltekintve Napster tulajdonképpen nem P2P hálózat

Miért nem skálázható Napster?

- Napster
  - Client-Server struktúra csillag topológiának felel meg
  - A gráf foka *n*-1
    - *n* a Peer-ek száma
  - A csillag csak 1-szeresen összefüggő
  - Egy gráf *k*-szorosan összefüggő, ha
    - bármely *k*-1 csomópont eltávolítása után még mindig összefüggő marad



- Napster nem skálázható, mert
  - a gráf foka nagy
    - szűk keresztmetszet okoz a kommunikációban
  - összefüggőség alacsony
    - nem robusztus a konstrukció

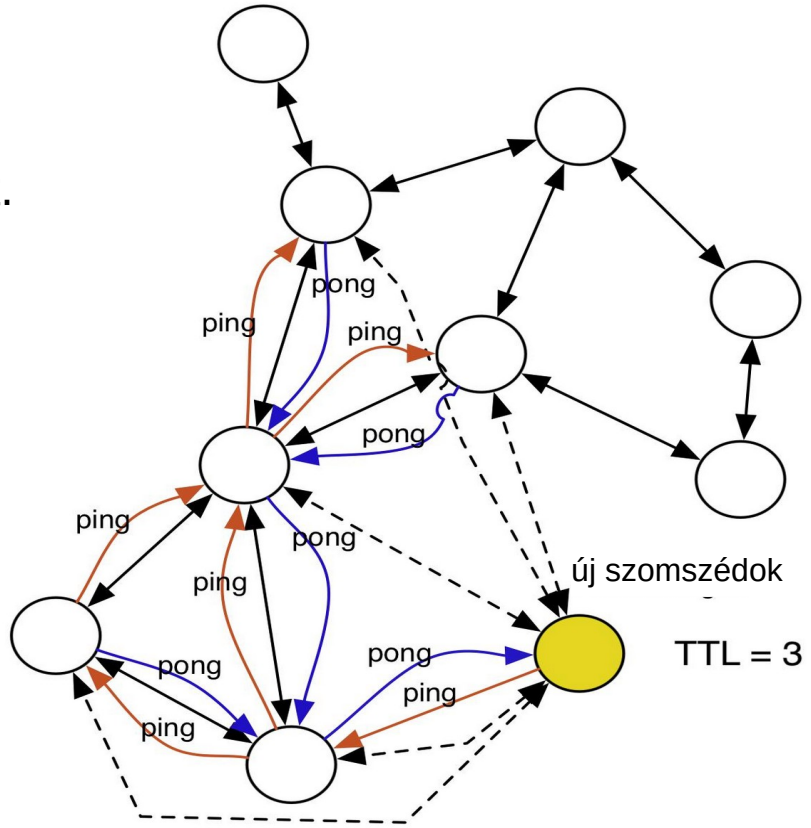


Gnutella - történet

- Gnutella
  - 2000 márciusában adta ki Justin Frankel és Tom Pepper a Nullsoft-tól
  - Nullsoft 1999 óta az AOL tulajdona
  - Cél: mint a Napster esetén, fájlmegosztás
  - Teljesen centrális struktúrák nélkül dolgozik

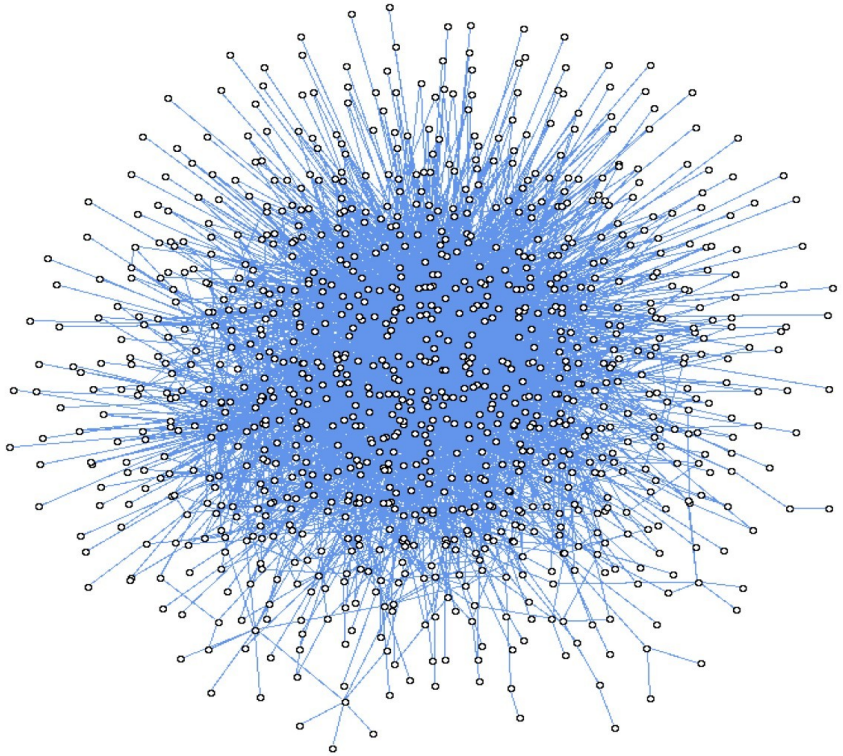
Gnutella – eredeti változat: kapcsolódás

- Szomszédság-listák
  - Gnutella kliens közvetlenül kapcsolódik más kliensekhez
  - Belépésnél az új kliensnek találni kell legalább egy másik csomópontot. (bootstrap nodes, gnutella web caches)
  - Ezek kipróbálásra kerülnek, amíg egy aktív jelentkezik
  - Egy aktív kliens ekkor továbbadja a szomszédság-listáját
  - A szomszádság-listákat a kliens mindig tovább hosszabbítja és eltárolja
  - Az aktív szomszédok száma korlátozva van (tipikusan 10-re)



Gnutella – eredeti verzió: kapcsolódás

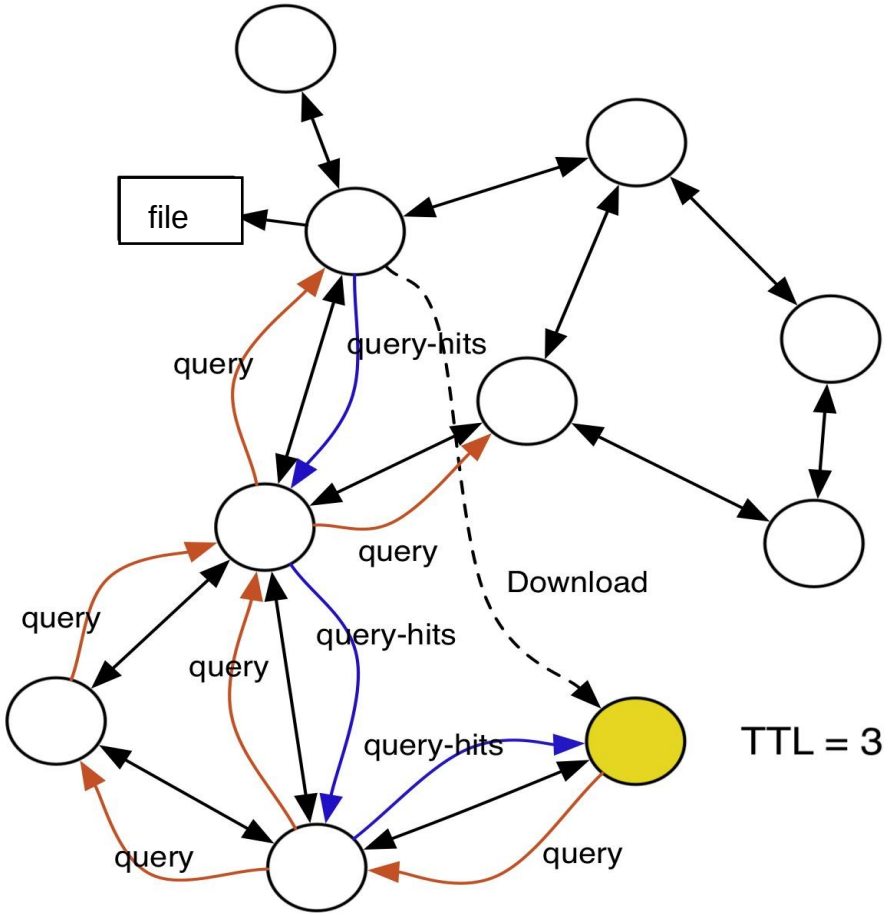
- Protokoll
  - `Ping`
    - Résztvétel-kérés
    - A TTL-nek (time to live) megfeleően továbbítódik
  - `Pong`
    - Reakció a Ping-re
    - A kérés útvonalán továbbítódik visszafele
    - Tartalmazza kérdezett résztvevő IP-jét és Port-ját
    - Tartalmazza a rendelkezésre bocsájtott fájlok számát és méretét
- Gráfstruktúra
  - Véletlen folyamat által képződik
  - Pareto-eloszlású
  - Felügyelet nélkül keletkezik



Gnutella Pilanatfelvétel 2000-ben

Gnutella – eredeti verzió: fájl lekérése

- Fájl kérés
  - a kérő elküldi minden szomszédjának
  - a szomszédok továbbküldik a saját szomszédjaiknak
  - egy adott mélységig
    - TTL (time to live)
- Protokoll
  - `Query`
    - A fájl kérése TTL-ben adott mélységig továbbítódik
  - `Query-hits`
    - Válasz ugyanazon az útvonalon visszafelé
- Ha fájlt megtaláltuk, közvetlen letöltés

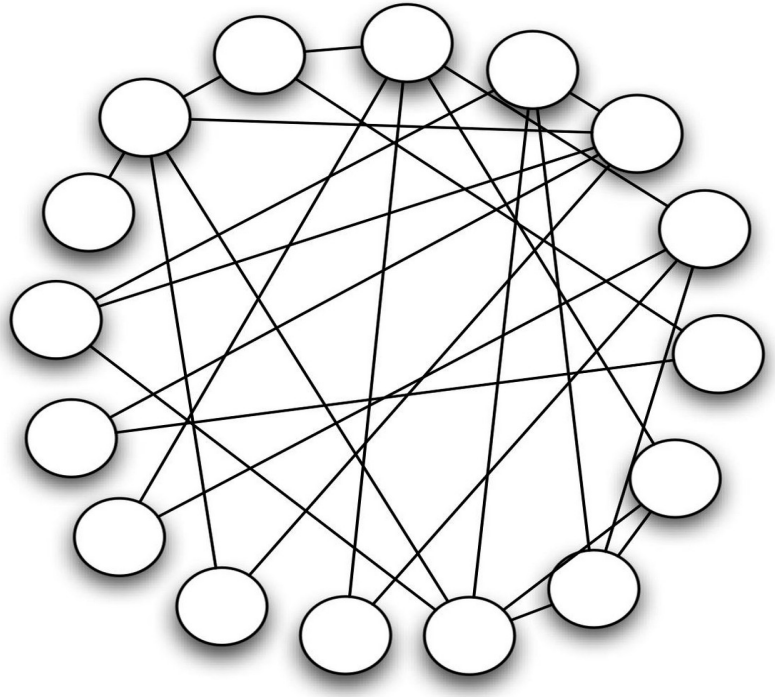


Gnutella - diszkusszió

- Előnyök
  - Elosztott
  - A hálózat skálázható
- Hátrányok
  - A kérések TTL-je által a hálózat implicit particionálódik
  - Ezáltal a kérések gyakran sikertelenek
  - A hosszú utak miatt, hosszú válaszidők
- Javítási javaslatok
  - Véletlen bolyongás a broadcast helyett
  - Az információk passzív replikálása az utakon

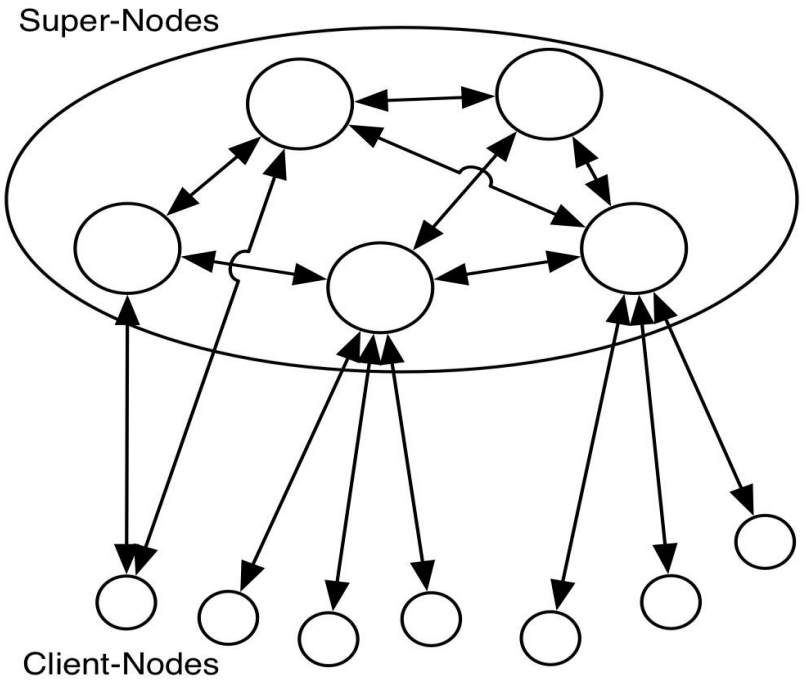
Miért nem skálázható Gnutella?

- Gnutella
  - Gráf- struktúra egy véletlen kapcsolat-gráf
  - A gráf foka kicsi
  - A gráf átmérője kicsi
  - Összefüggősége nagy
- A keresés erőforrásigényes
  - ahhoz, hogy biztosan megtaláljunk egy adatot, az egész hálózatot át kell keresni
- Gnutella nem skálázható, mert
  - nincs struktúrája az adatok tárolásának



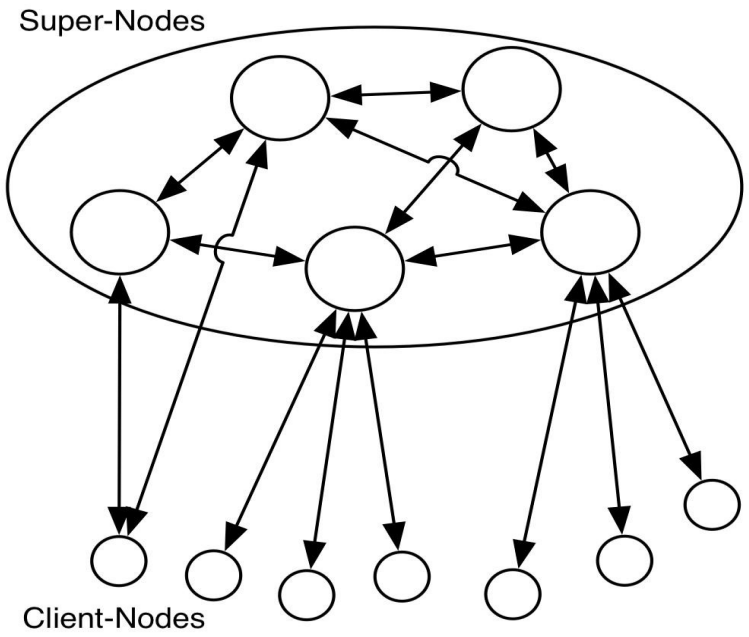
Kazaa, Gnutella (II), Morpheus

- Hibrid Struktúra
  - Nagy sávszélességű csomópontokat kiválaszt P2P-szervernek (super-nodes)
  - Ezek a csomópontok egy Gnutella stílusú P2P hálózatot alkotnak
  - Normális csomópontokat ezekhez a szuper-csomópontokhoz kapcsolja a hálózat kliesként
- Ilyen hálózatot valósít meg
  - Kazaa
  - Morpheus
  - Gnutella (új verzió)
- Előnyök
  - Jobban skálázható
  - Rövidebb válaszidők
- Hátrányok
  - A kliensek megtagadhatják a szuper-csomópontnak való kiválasztást



Miért nem skálázható Kazaa és társai?

- Hibrid struktúra
  - Átmérő kicsi
  - Az összefüggőség választható nagyra
    - a klienseknél a szuper-csomópontok száma által
  - A fokszám kicsi
- Skálázhatóság
  - nem olyan rossz, mint a Gnutella-nál vagy a Napster-nél
  - minden szuper-csomópont, megkapja a kliensek kéréseit



Elvárások P2P hálózatoknál

- Kezelhetőség
- Információ koherencia
- Hibatűrés
- Biztonság
- Skálázhatóság

Struktúra nélküli / struktúrált P2P hálózatok

Adat visszakeresés:

- Hol van?
- Hogy jutunk oda?

Struktúra nélküli hálózatok:

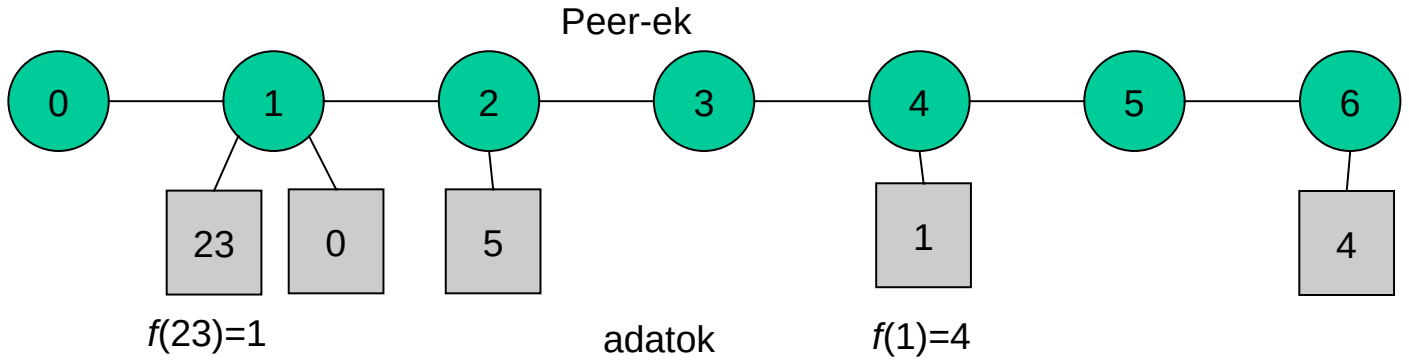
- Napster:
  - Hol?
    - A szerveren ☺
  - Hogy jutunk oda?
    - torlódás/dugó a szerveren ☹
- Gnutella
  - Hol?
    - Nem tudjuk ☹
  - Hogy jutunk oda?
    - Mindenkit megkérdezünk ☹

Struktúrált hálózatok:

- Hol van az *x* adat?
  - Az *f*(*x*) helyen
  - Mi az az *f*(*x*)?
    - *x* egy minden résztvevő számára ismert leképezése egy térre
- Hogy jutunk oda?
  - Egy jól definiált útvonalon, amely a kérdező helyétől *f*(*x*) helyréhez.

Egy hash-tábla mint P2P hálózat

- Minden Peer egy tárhelyet reprezentál 0,1,2,..,*n*-1
  - *f*(*x*): egy minden Peers számára ismert hash-függvény, pl. *n* = 7 esetén
    - *f*(*x*) := (3*x*+1 mod 23) mod 7
  - A Peer-eket kössük össze láncként



- Keresés
  - Számítsuk ki *f*(*x*)-et
  - Menjünk oda ahhoz a Peer-hez a láncon, amely *f*(*x*)-et reprezentálja

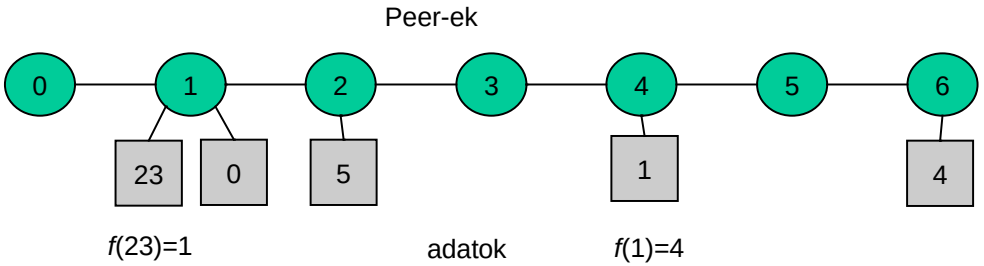
Hash-táblától elosztott hash-táblához (DHT)

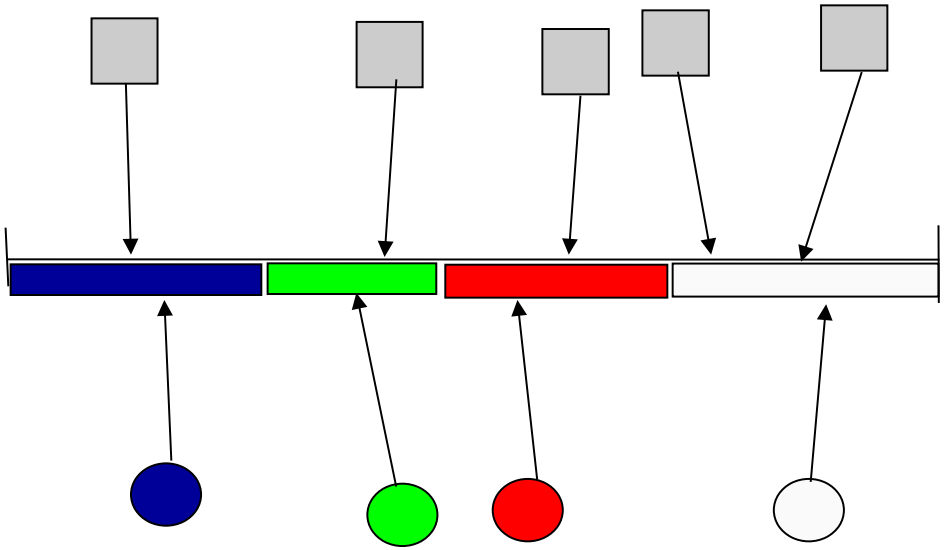
Hash-Táblák

- Előny
  - A keresés egyszerű
- Hátrány
  - Egy új Peer kapcsolódásakor új hash-függvényt kell választani
  - Hosszú utak, nagy élterhelés

Elosztott Hash-Tábla (Distributed Hash Table, DHT)

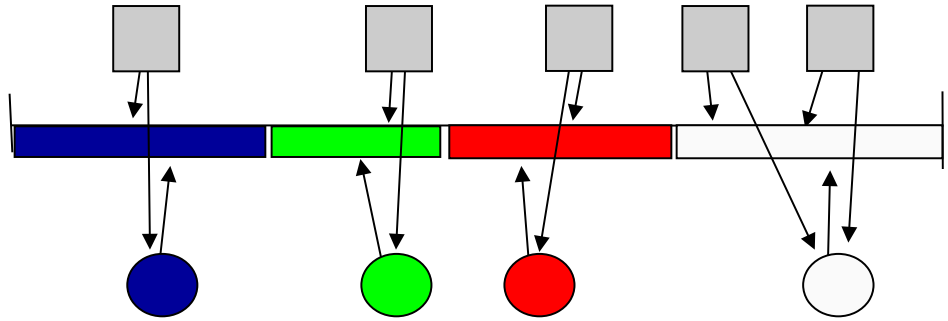
- A Peer-eket leképezzük (hashing által) egy helyre és minden Peer-hez hozzárendeljük a hash-függvény értéktartományának egy részét
- Az adatokat is hash-eljük
  - A hash-függvény értéké alapján a tartományért felelős Peer-en tároljuk



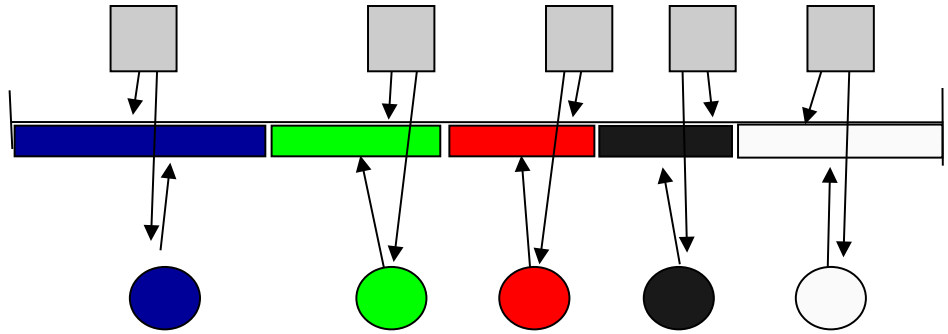


Befűzés a DHT-ba

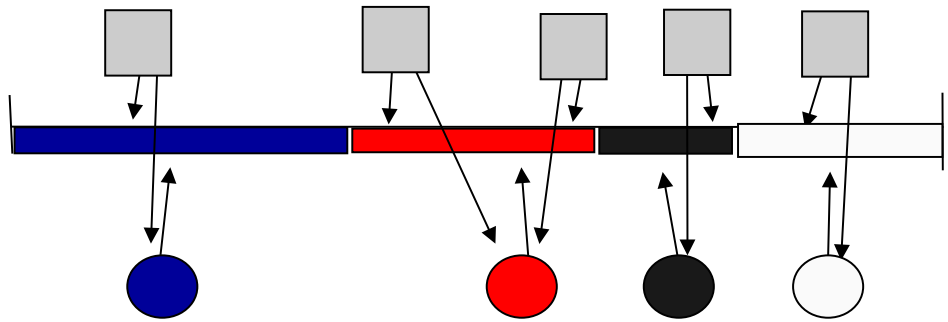
- Elosztott Hash-Tábla
  - Peer-ek et hash-eljük egy helyre
  - Mindegyik egy tartományért felelős
  - Adatokat szintén hash-eljük



- Bekapcsolódik egy új Peer (csomópont)
  - A szomszédok átadják a tartományuk egy részét és a hozzátartozó adatokat

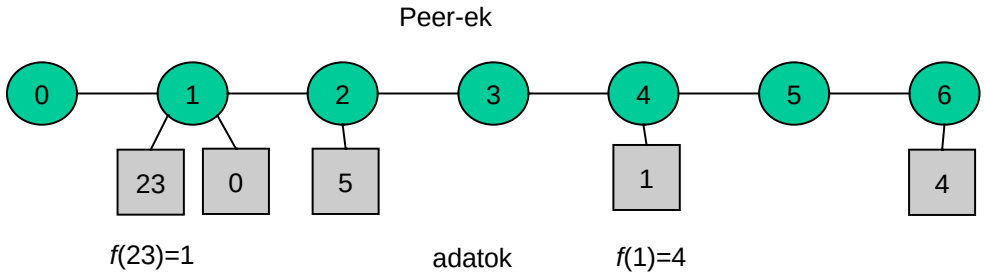


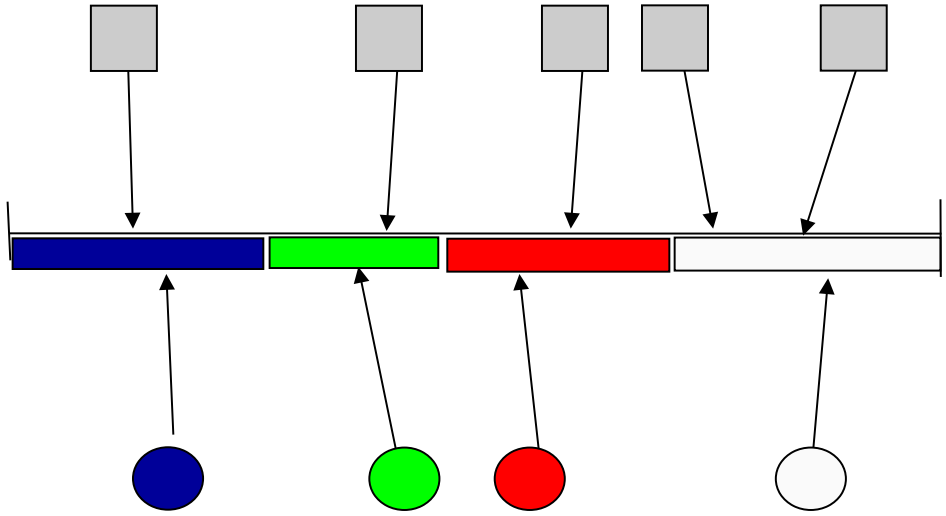
- Egy Peer elhagyja a hálózatot
  - A szomszédai átveszik a tartományát és a hozzátartozó adatokat



DHT tulajdonságai

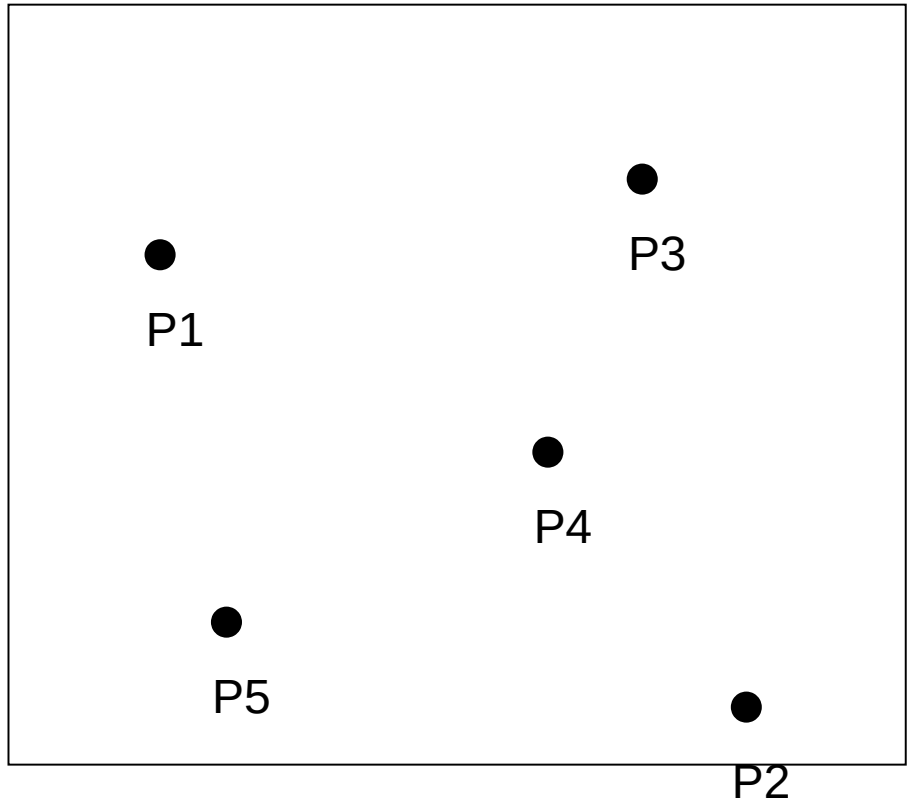
- Előny:
  - Minden adatot egyértelműen hozzá lehet rendelni egy Peer-hez
  - Egy Peer csatlakozása vagy kilépése a hálózatból csak a szomszédainál okoz változást
- DHT-t sok P2P hálózat használ
- Még tisztázni kell:
  - A kapcsolódások struktúráját





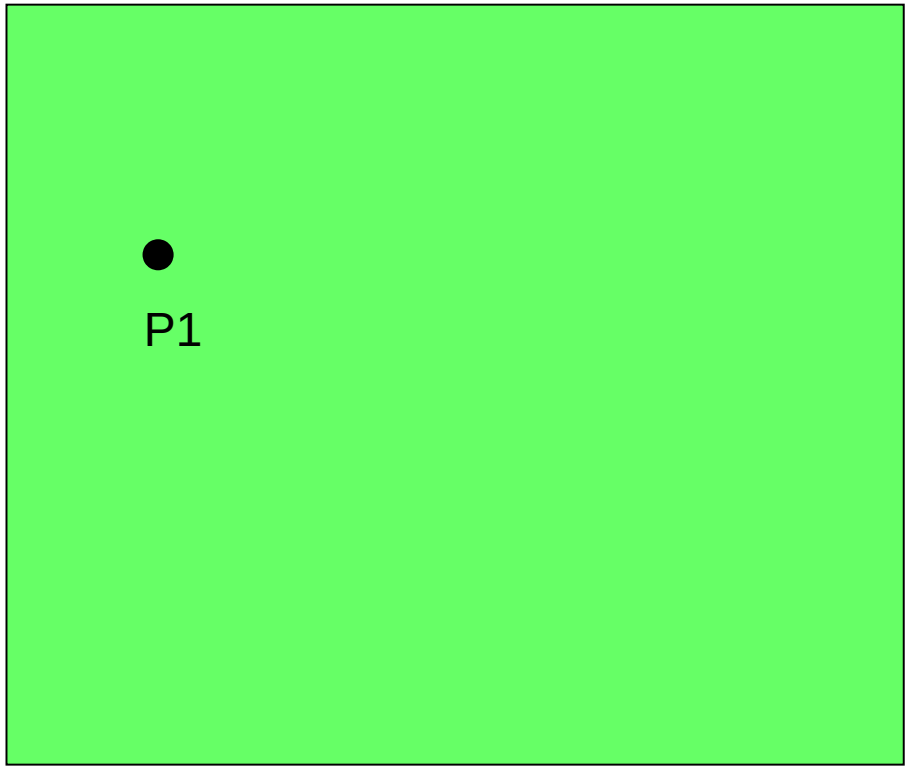
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek



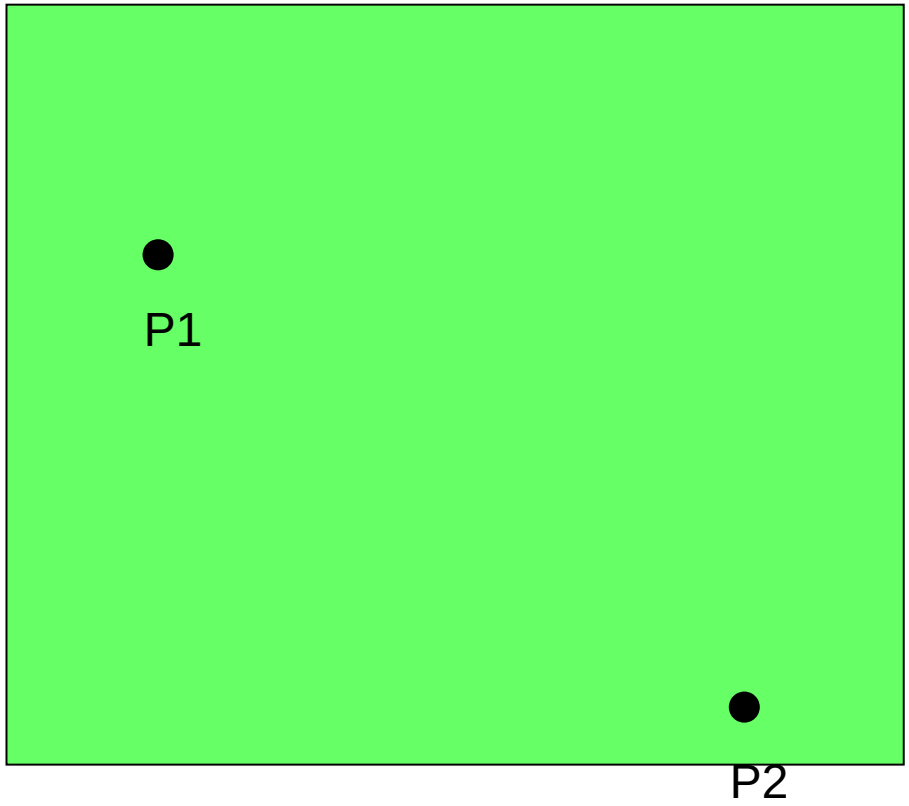
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)



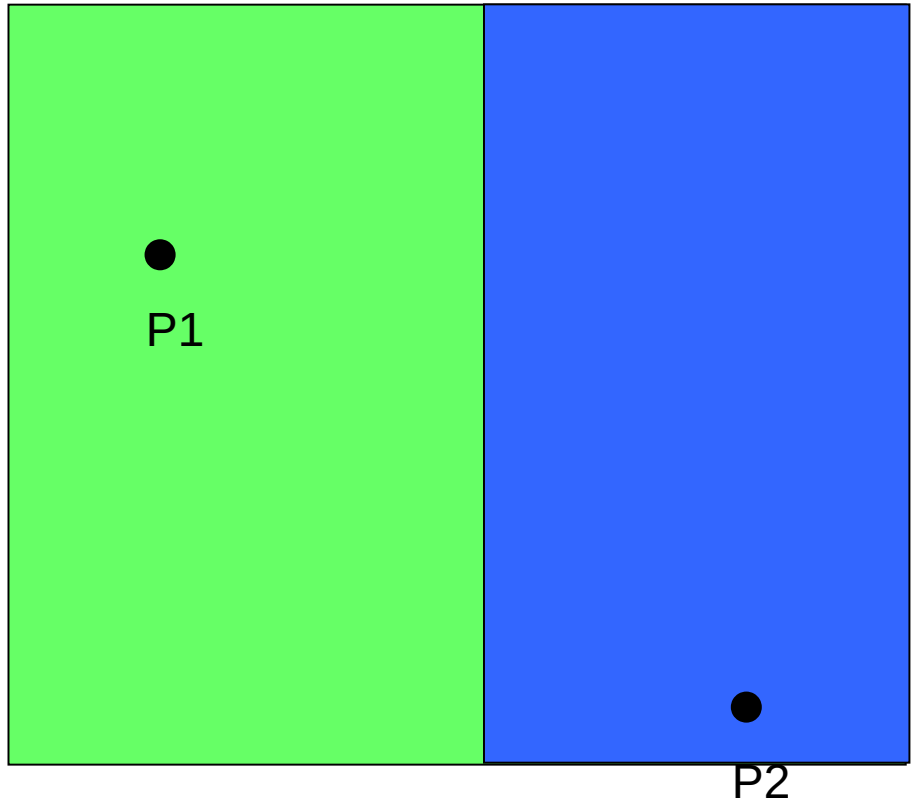
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



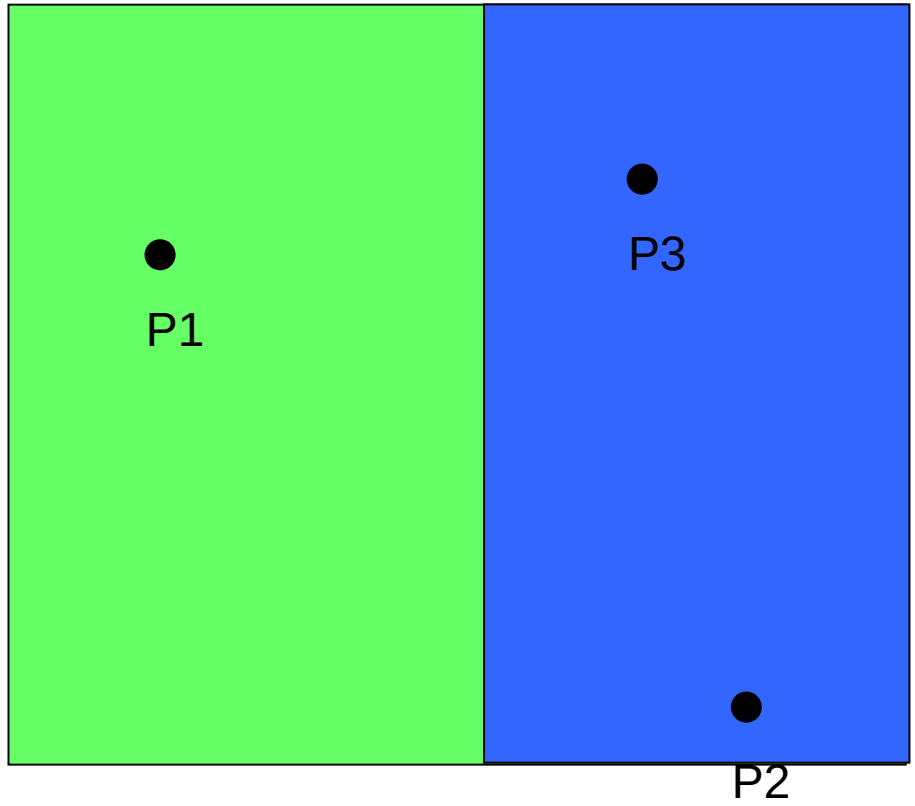
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



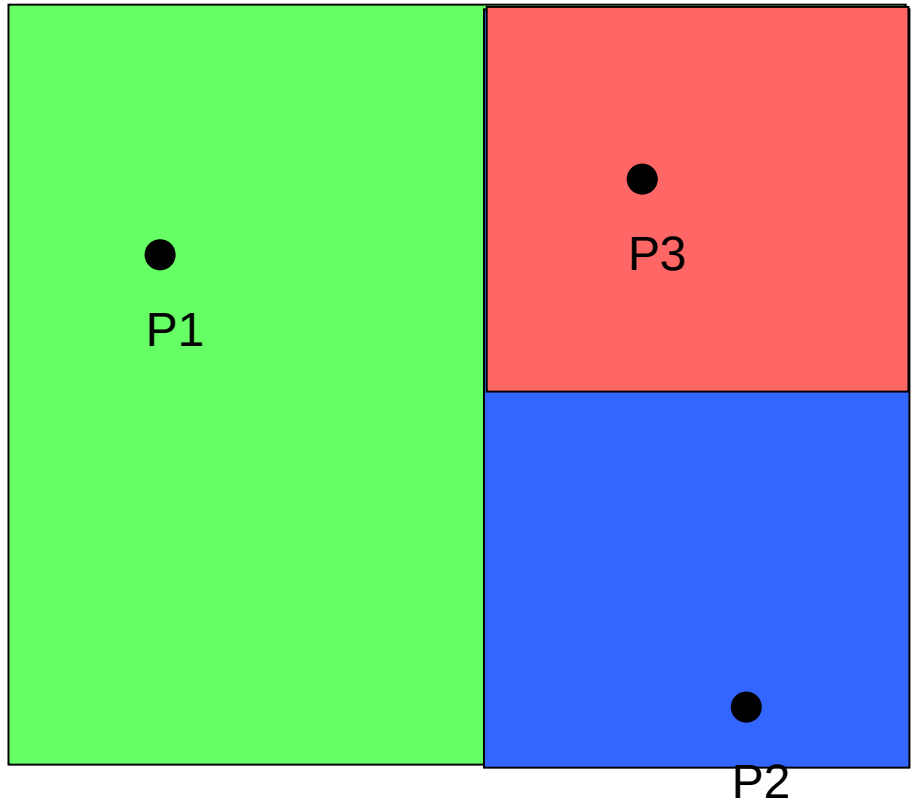
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



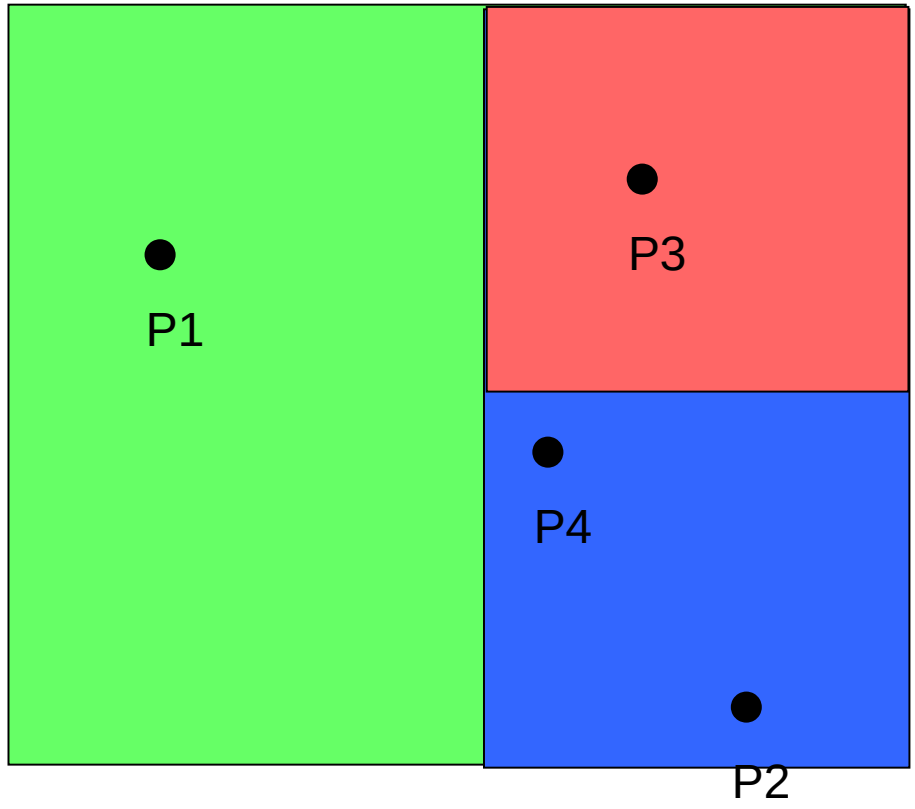
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



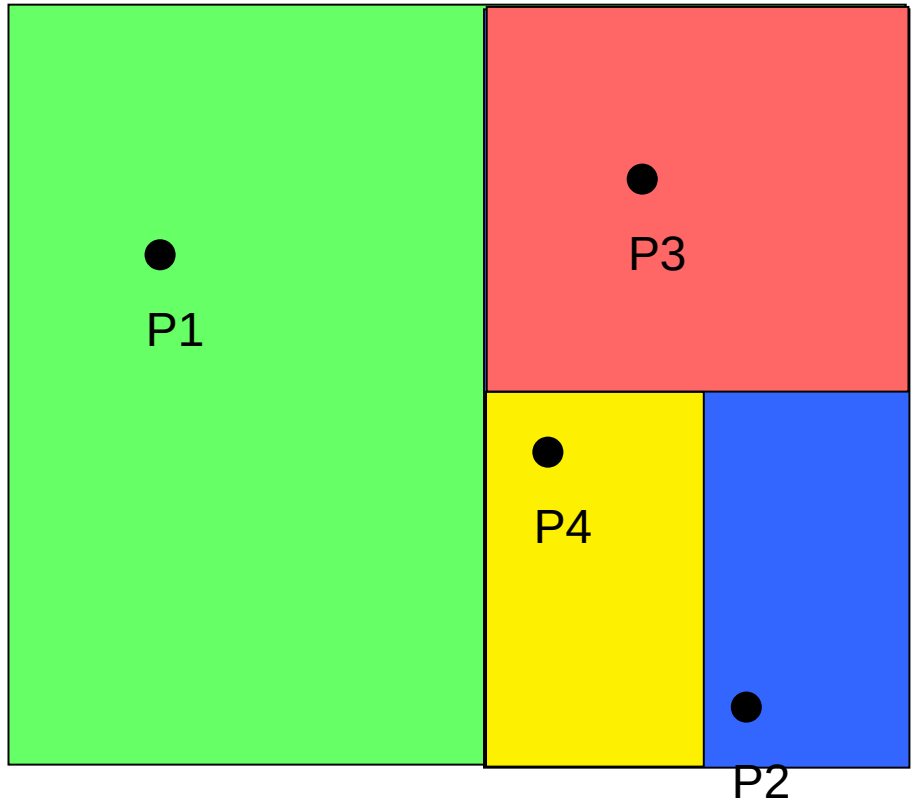
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



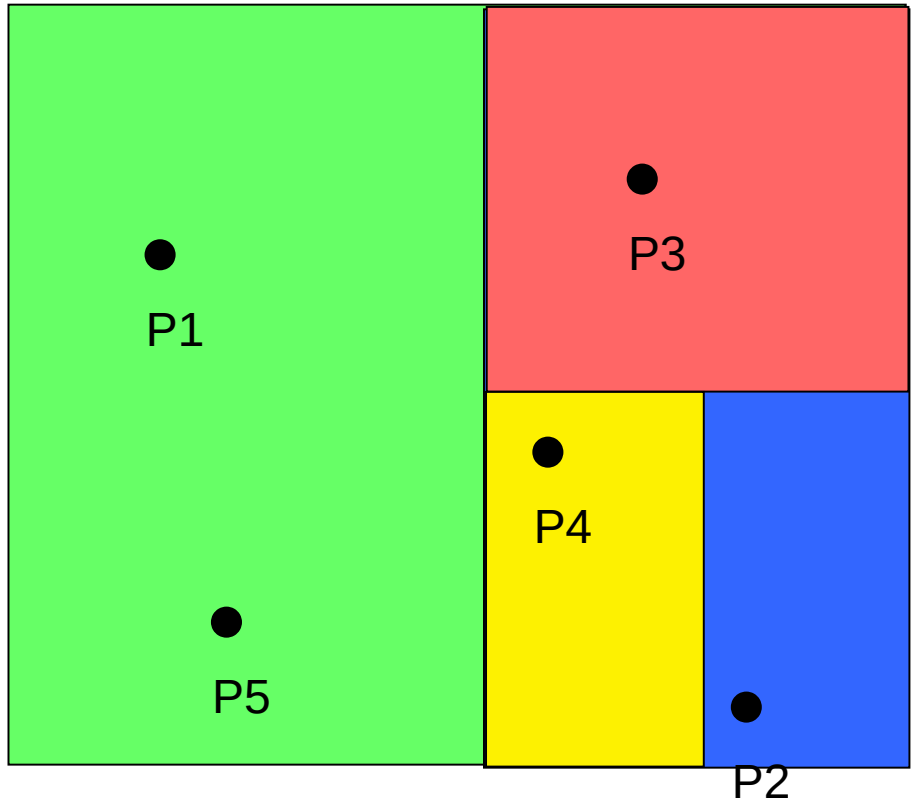
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



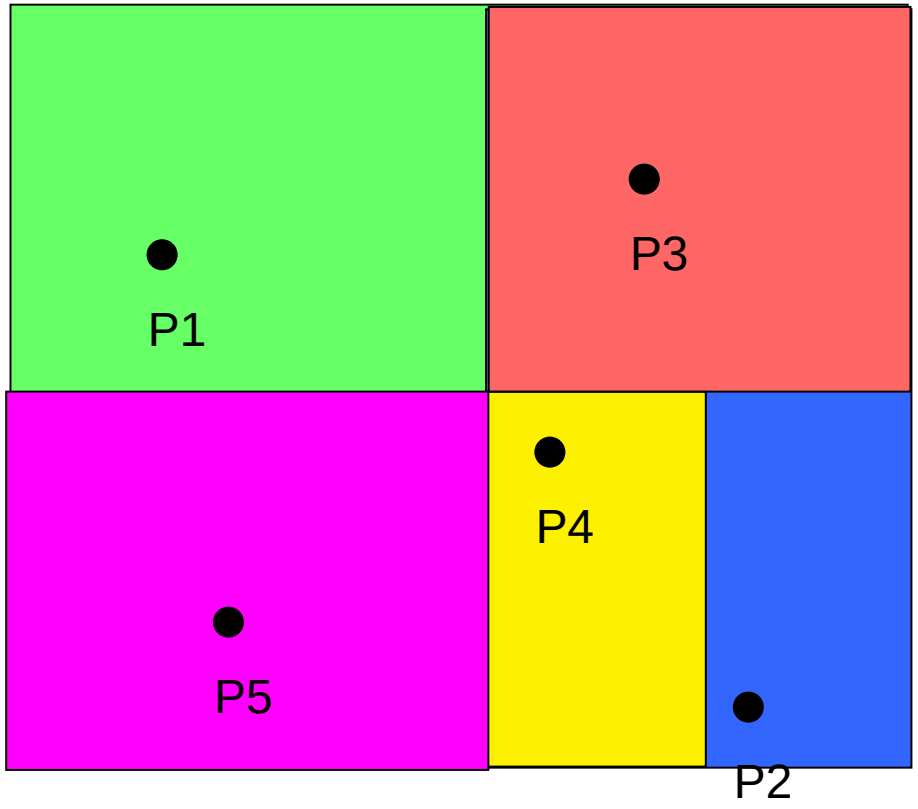
Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek



Content Addressable Network (CAN)

- A Peer-ek és a fájlok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos
  Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ egy véletlen pontot a négyzetben (hash-függvény)
  - A megfelelő négyszög tulajdonosa kettéosztja a négyszöget és
  - átadja a felét az új Peer-nek

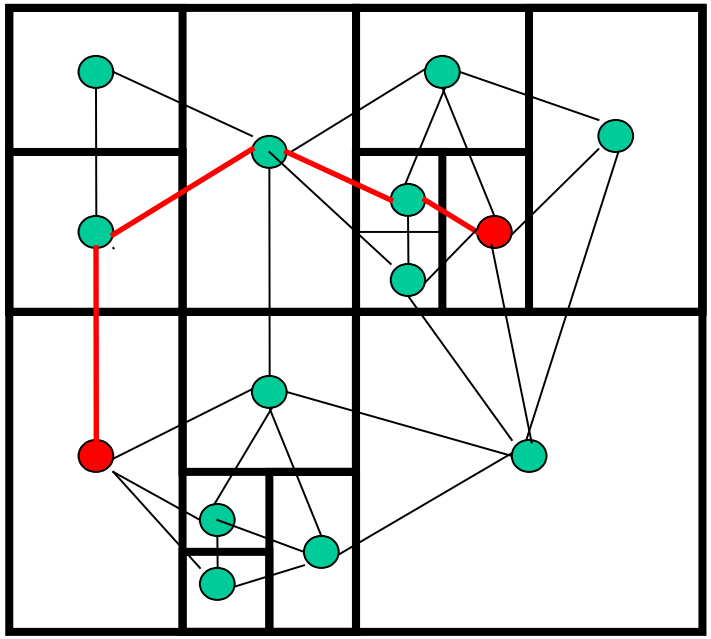


Keresés a CAN-ban

- Először az adat helyét határozzuk meg a hash-függvény értékének kiszámolásával
- A szomszédos négyszögek tulajdonosai között élek vannak
- A kérés az adat helyének irányába továbbítódik

- *d* dimenziós négyzet/kocka
  - 1 dimenziós: szakasz
  - 2 dimenziós: négyzet
  - 3 dimenziós: kocka
  - 4: ...

- Az élek várható értéke az úton: $n^{1/d}$
- A csomópontok átlagos fokszáma: O(*d*)



Irodalom

- https://computer.howstuffworks.com/napster.htm
- D. Karger, E. Lehman. T. Leighton, R. Panigrahy, M. Levine, and D. Lewin: **Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web.** In *Proc. 29th STOC*, 654-663, 1997.
- M. Jovanovic, F. Annexstein, and K. Berman: **Scalability Issues in Large Peer-to-peer networks: A case study of Gnutella**.
- S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker: **A scalable content-addressable network**. In *Proc. ACM SIGCOMM*, 161-172, 2001.