



# Hálózatok II 2007

## 11 Peer-To-Peer Hálózatok II

## P2P hálózatok kritériumai

- Kezelhetőség
  - Milyen nehéz a hálózatot működtetni
- Információ-koherencia
  - Mennyire jól osztja el az információt?
- Bővíthetőség
  - Milyen könnyen tud növekedni?
- Hibatűrés
  - Milyen könnyen hárítja el a hibákat?
- Biztonság
  - Mennyire nehezen rombolható szét tudatosan?
- Skálázhatóság
  - Milyen nagyra tud a hálózat növekedni?

# Content Addressable Network (CAN)

Két kérdés az információ keresésénél

- Hol van?
- Hogy jutunk oda?

● Napster:

- Hol?
  - A szerveren 😊
- Hogy jutunk oda?
  - torlódás/dugó a szerveren ☹️

● Gnutella

- Hol?
  - Nem tudjuk ☹️
- Hogy jutunk oda?
  - Mindenkit megkérdezzük ☹️

● Egy jobb ötlet:

● Hol van az  $x$  adat?

● Az  $f(x)$  helyen

● Mi az az  $f(x)$ ?

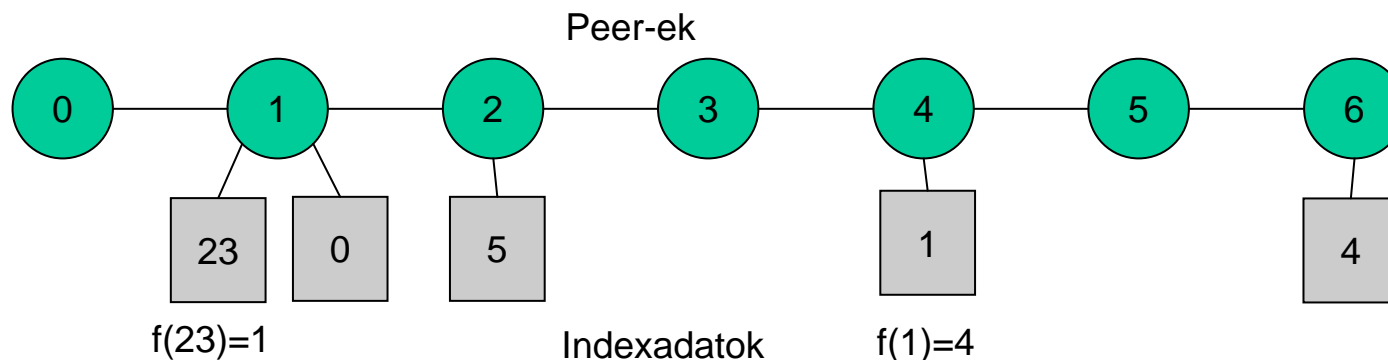
●  $x$  egy minden résztvevő számára ismert leképezése egy térre

● Hogy jutunk oda?

● Egy jól definiált útvonalon, amely a kérdező helyétől  $f(x)$  helyréhez vezet.

# Egy hash-tábla mint Peer-to-Peer hálózat

- Minden Peer egy tárhelyet reprezentál  $0, 1, 2, \dots, n-1$ 
  - $f(x)$ : egy minden Peers számára ismert hash-függvény, pl.  $n = 7$  esetén
    - $f(x) := (3x+1 \bmod 23) \bmod 7$
  - 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

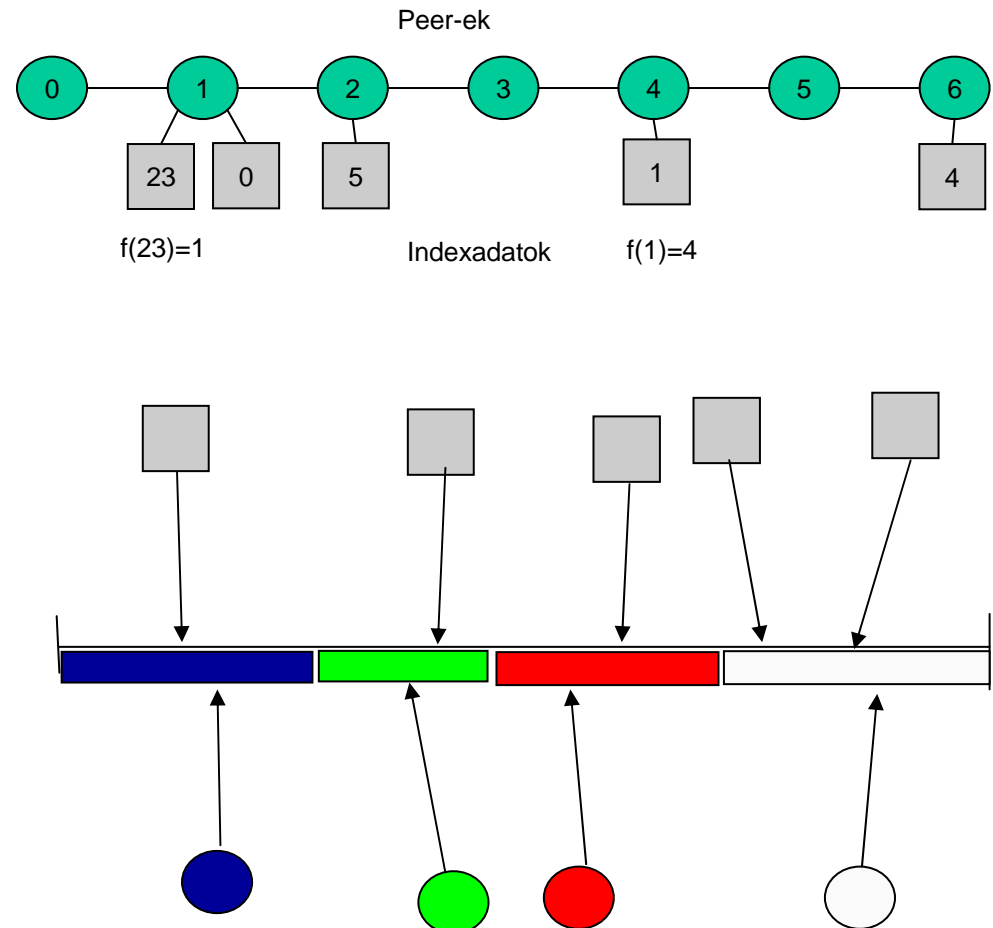
# Hash-táblától az elosztott hash-táblához (DHT)

## Hash-Táblák

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

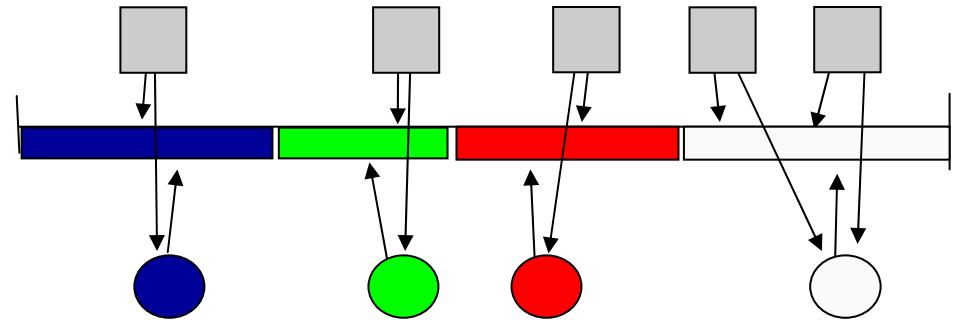
## Elosztott Hash-Tábla (Distributed Hash Table, DHT)

- 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

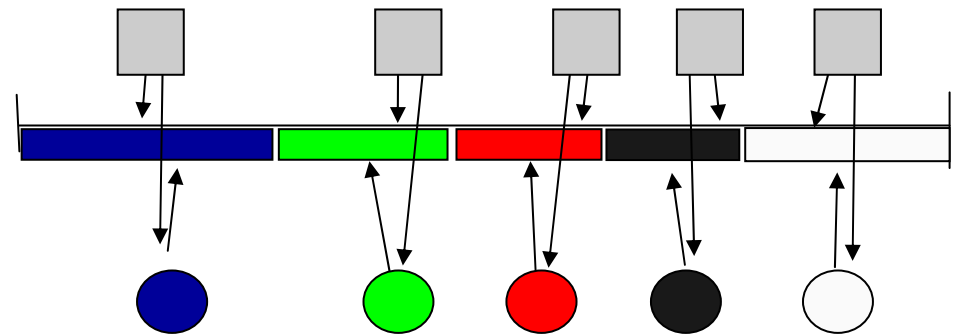


# Befűzés a DHT-ba

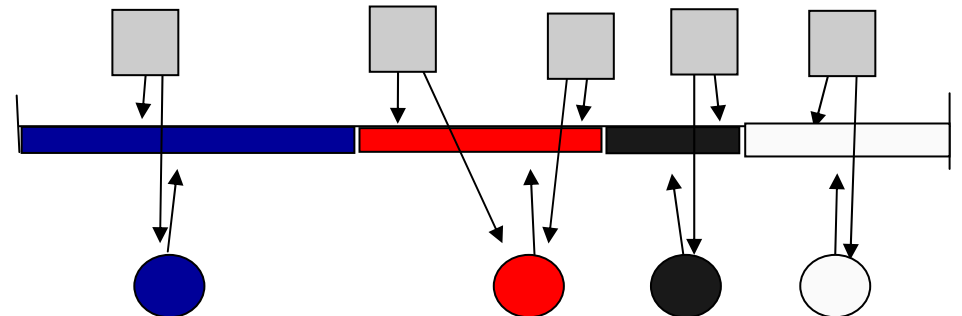
- Elosztott Hash-Tábla
  - Peer-ek et hash-eljük egy helyre
  - Mindegyik egy tartományért felelős
  - Adatokat szintén hash-eljük



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

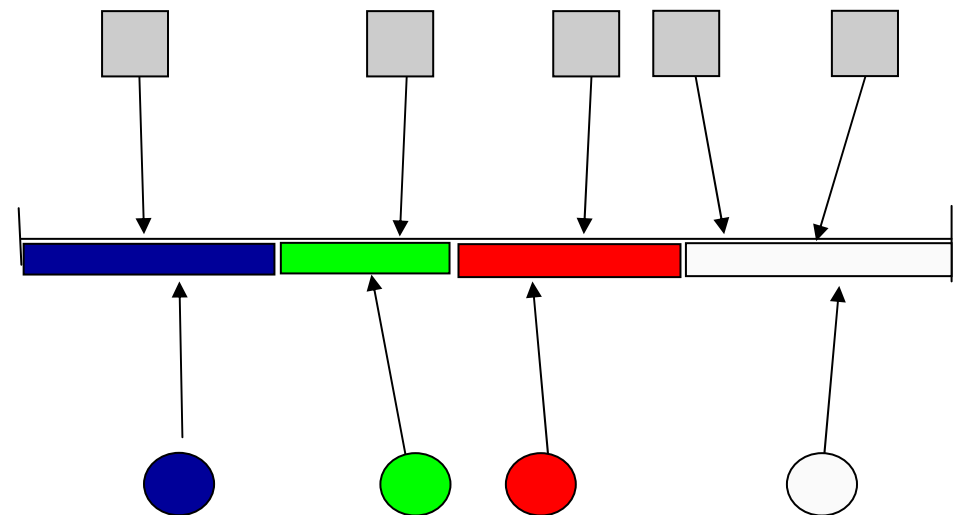
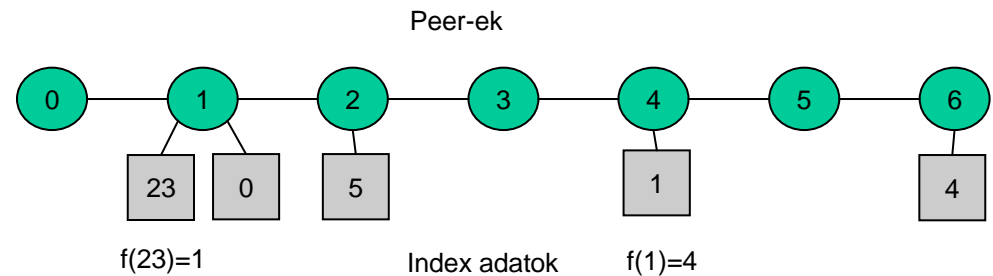


- Egy Peer elhagyja a hálózatot
  - A szomszédai átveszik a tartományát és a hozzátartozó adatokat



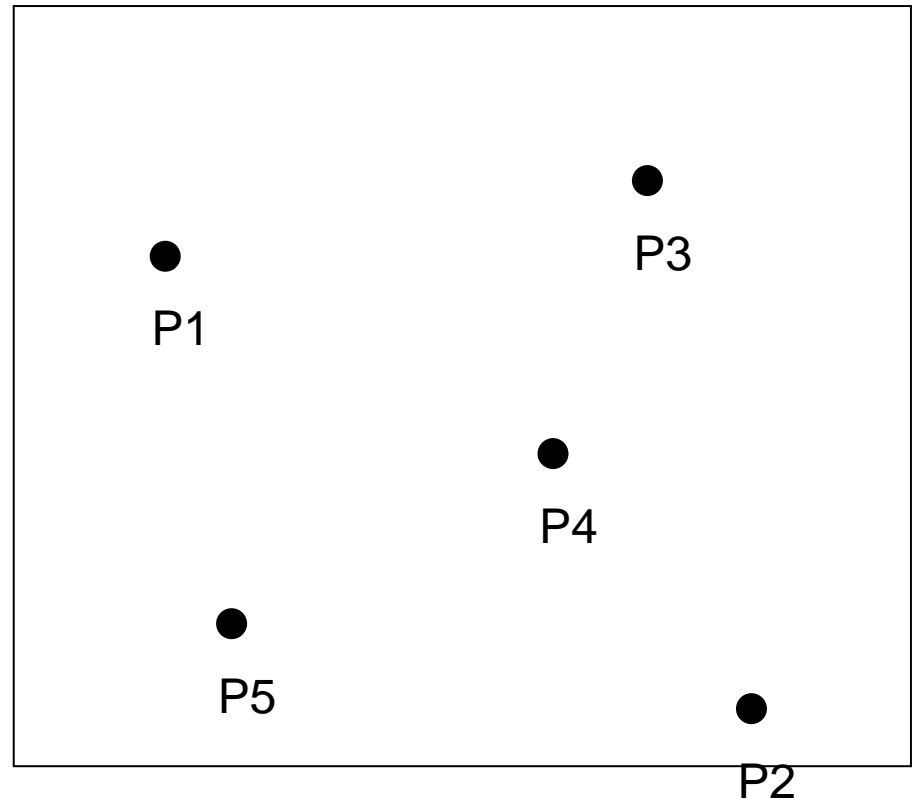
# DHT tulajdonságai

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



# Content Addressable Network (CAN)

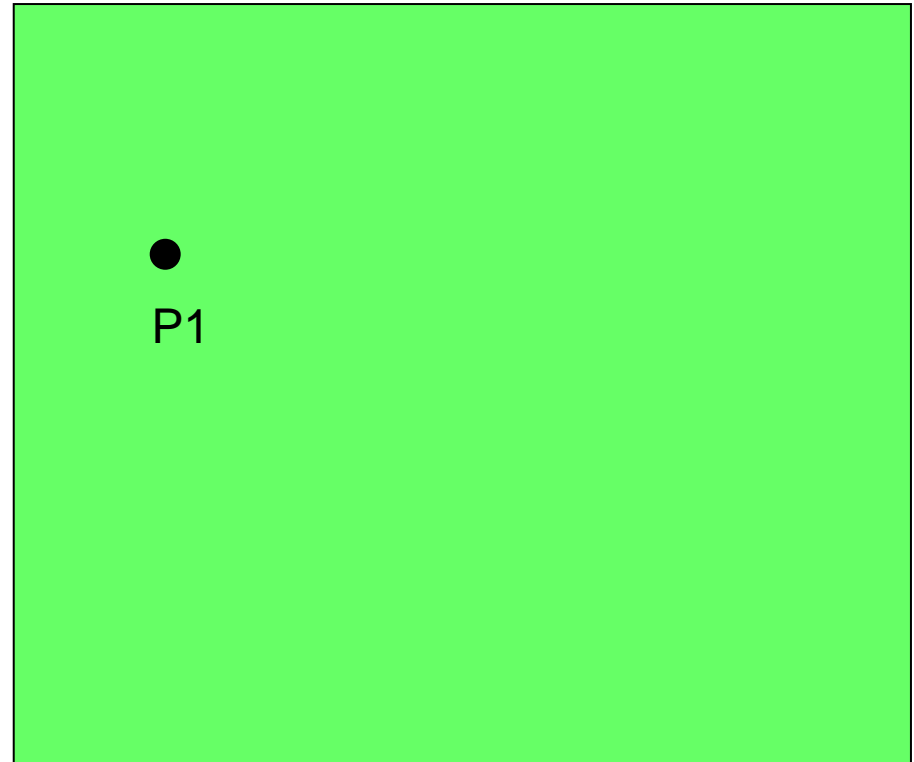
- A Peer-ek és a file-ok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek





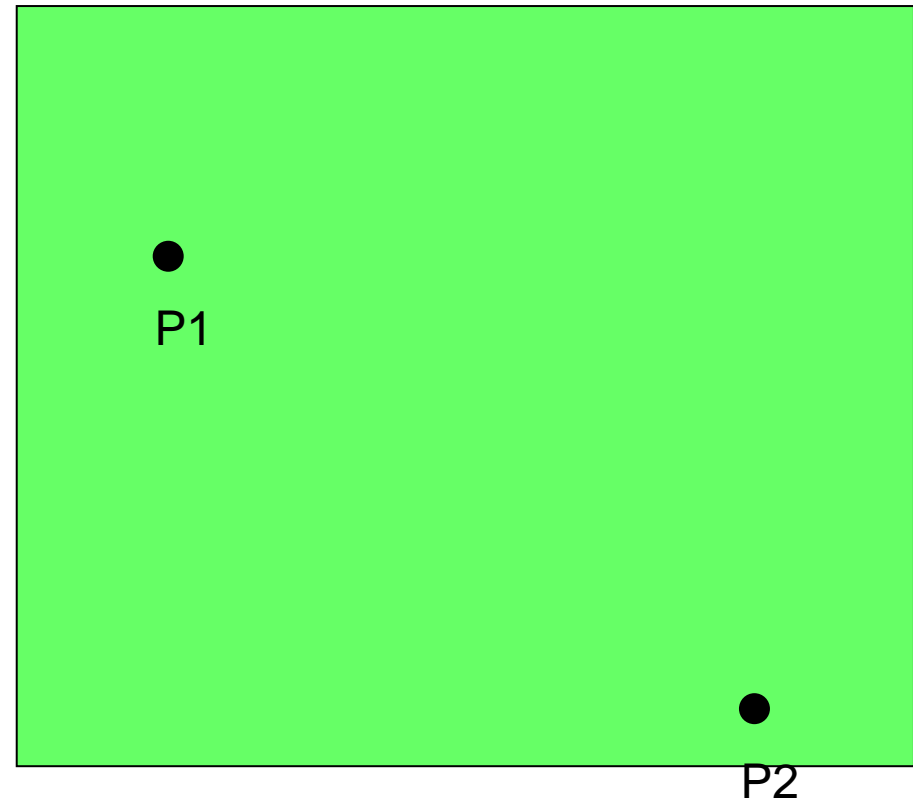
# Content Addressable Network (CAN)

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



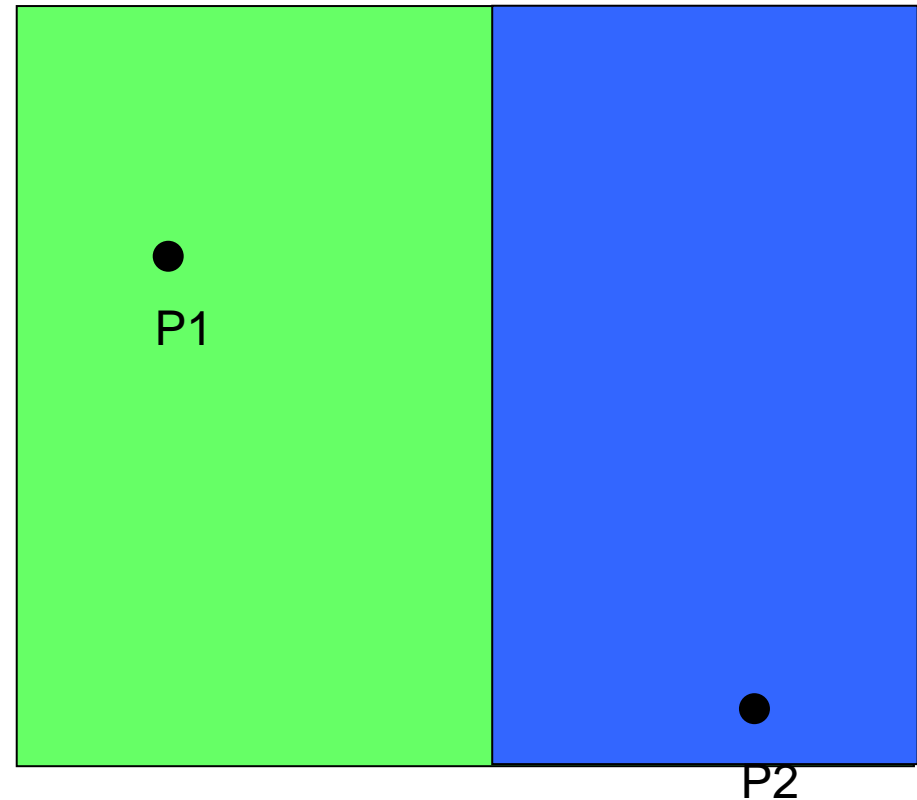
# Content Addressable Network (CAN)

- A Peer-ek és a file-ok 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



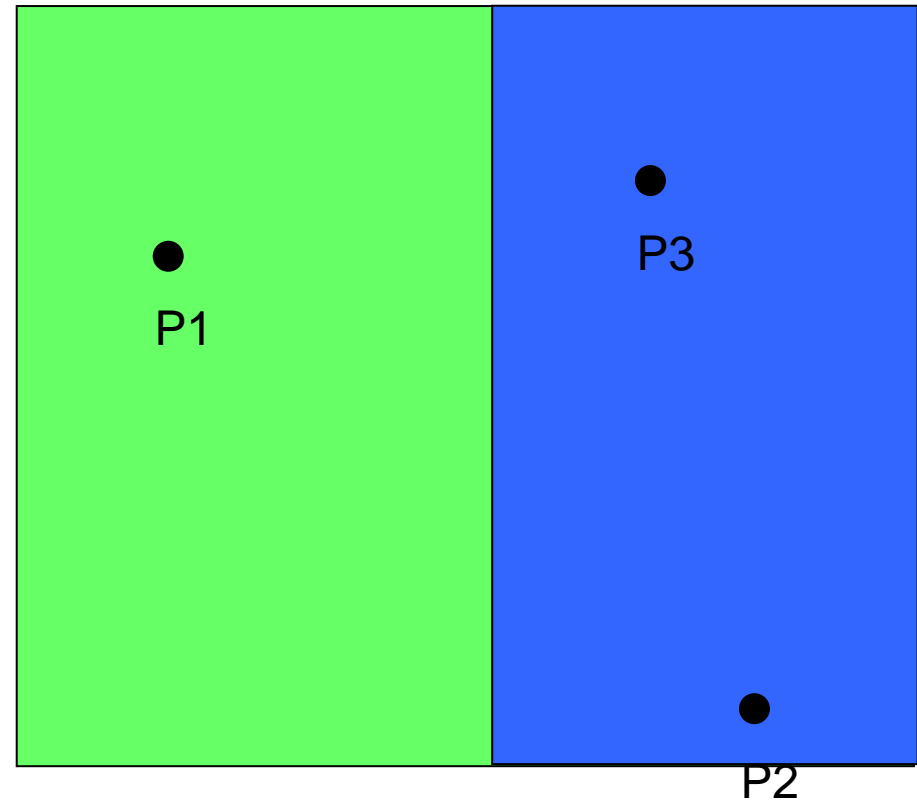
# Content Addressable Network (CAN)

- A Peer-ek és a file-ok 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



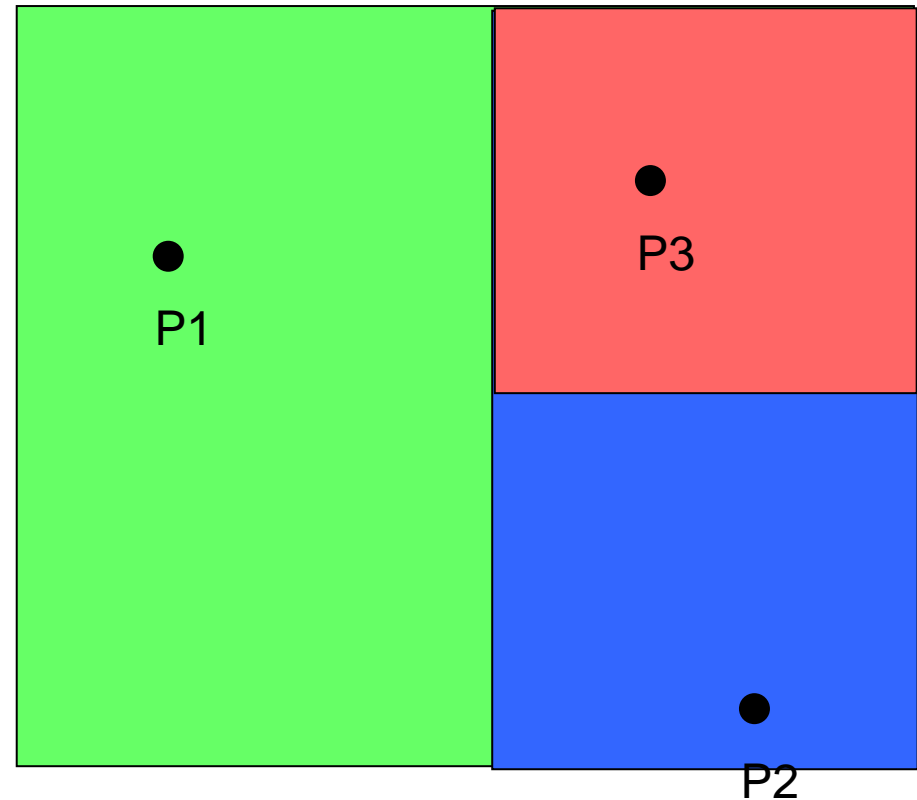
# Content Addressable Network (CAN)

- A Peer-ek és a file-ok 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



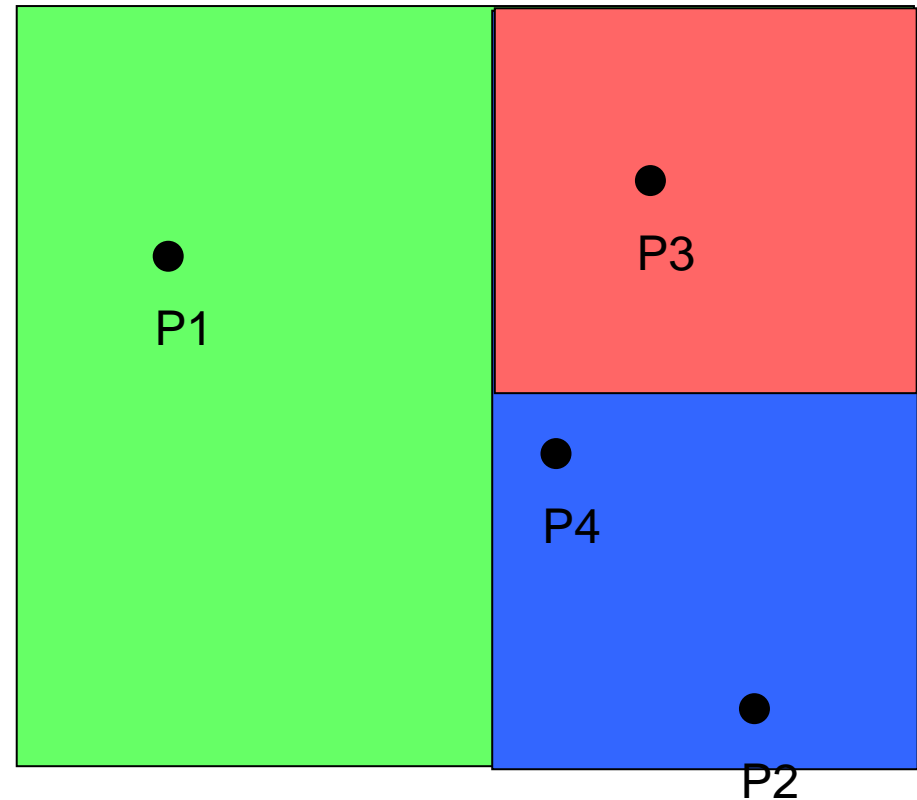
# Content Addressable Network (CAN)

- A Peer-ek és a file-ok 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



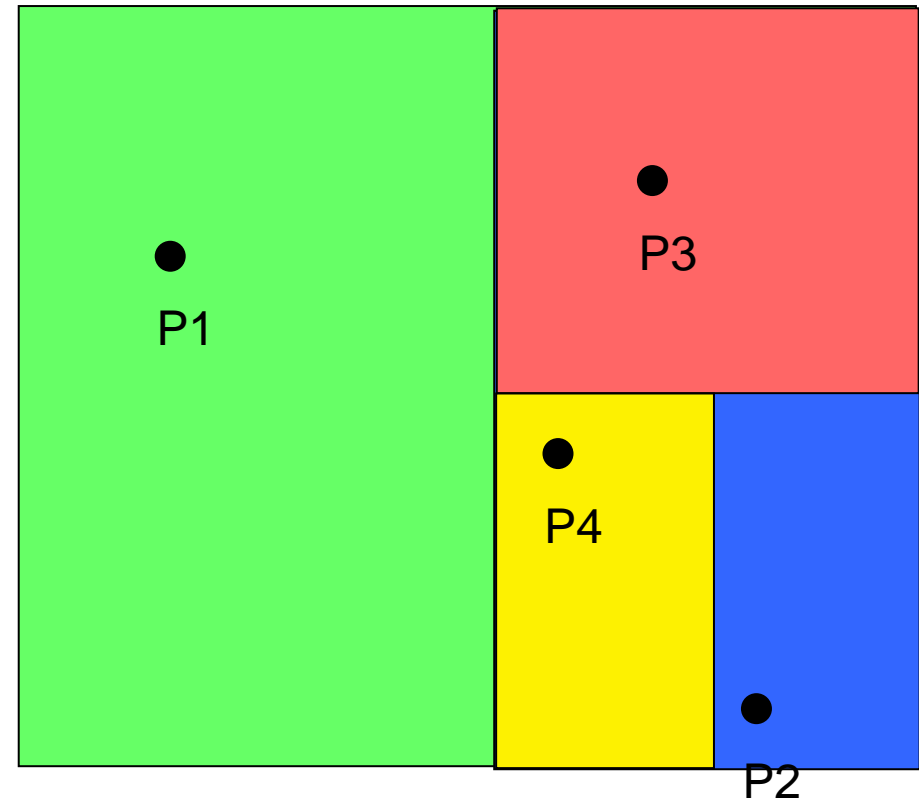
# Content Addressable Network (CAN)

- A Peer-ek és a file-ok 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



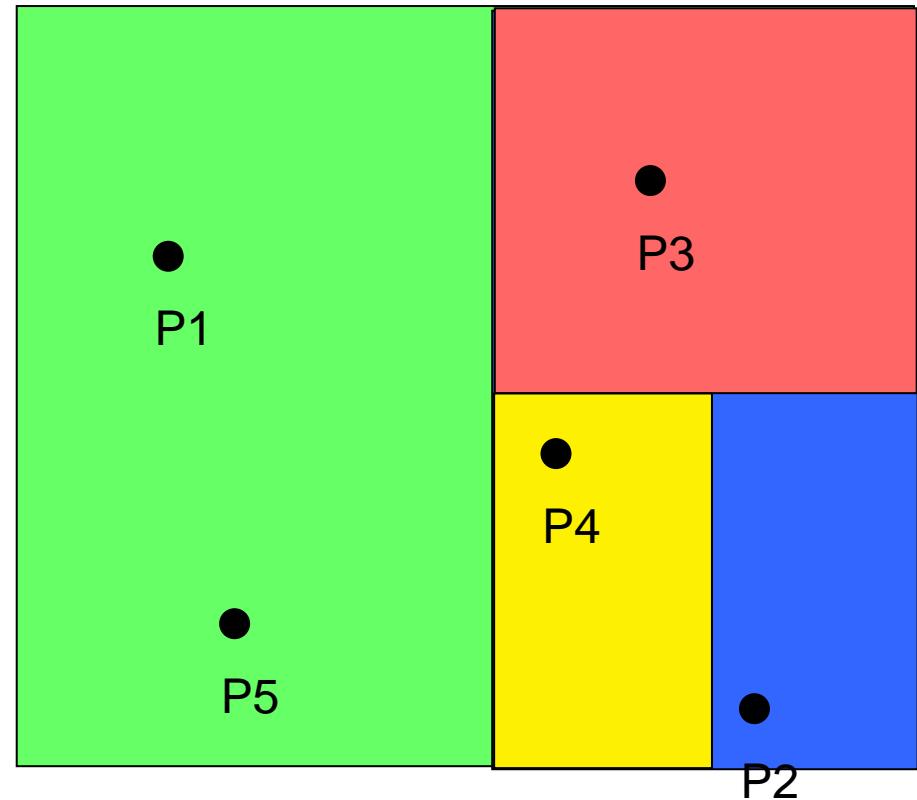
# Content Addressable Network (CAN)

- A Peer-ek és a file-ok 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)

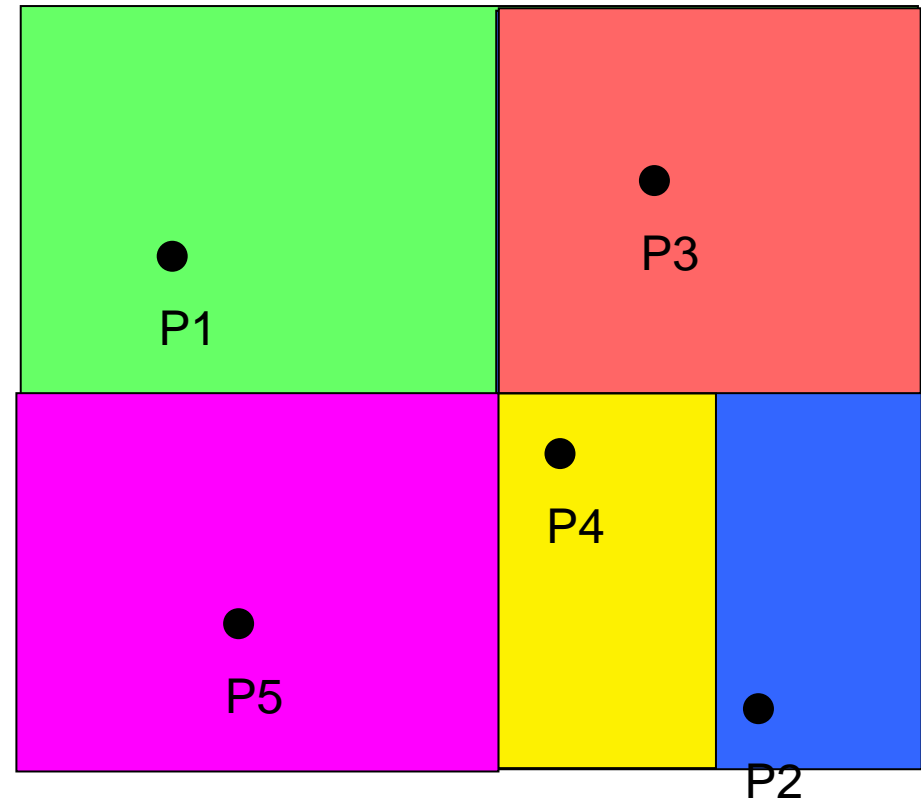
- A Peer-ek és a file-ok egy (kétértékű) hash-függvénnyel az egységnégyzetbe képeződnek
- Kezdetben egy üres négyzet és egyetlenegy Peer mint tulajdonos  
Ez a négyzet a Peer zónája (tartománya)
- Egy tartomány tulajdonosa minden adatot tárol, amely arra a tartományra képeződik le
- Egy Peer választ 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 file-ok 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



# Milyen nagyok/kicsik lehetnek a négyszögek

$R(p)$  : Egy  $p$  Peer négyszöge

$A(p)$  : Egy  $p$  Peer négyszögének területe

$n$  : Peer-ek száma

Kezdeti négyszög: terület 1

## Lemma

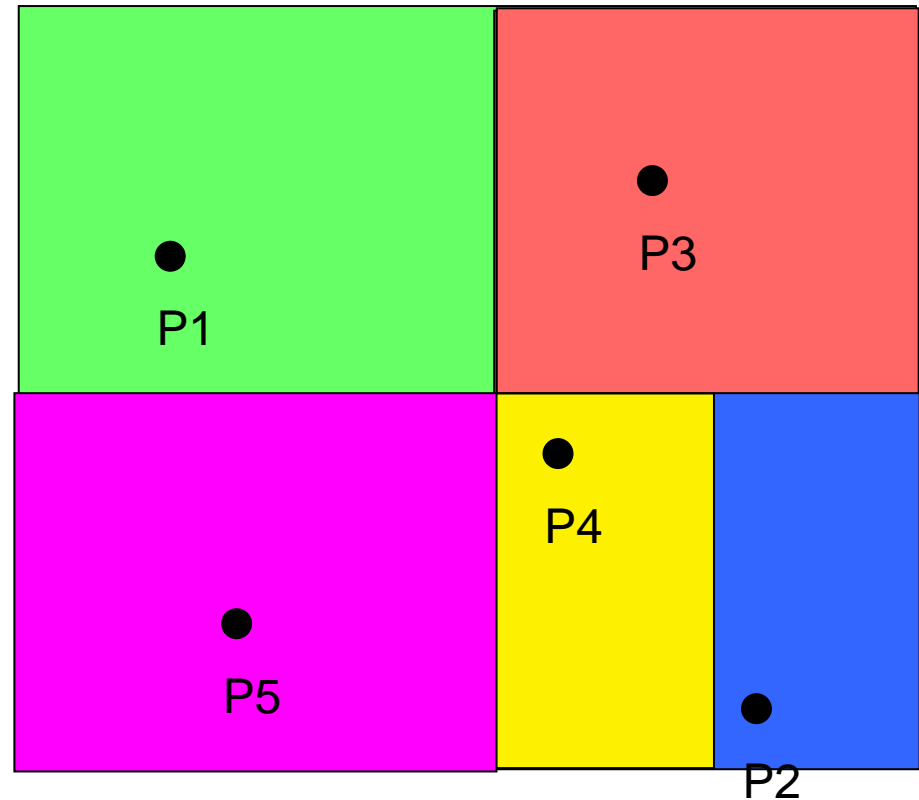
Minden  $p$  Peer-re érvényes

1.  $E[A(p)] = \frac{1}{n}$

2. Legyen  $P_{R,n}$  annak a valószínűsége, hogy  $n$  Peer közül egy se esik az  $R$  négyszögbe. Ekkor

$$P_{R,n} \leq e^{-n \text{Vol}(R)}$$

ahol  $\text{Vol}(R)$  az  $R$  területe



## Egy Peer négyszögének várható területe a CAN-ban

Biz. 1.-hez  $E[A(p)] = \frac{1}{n}$

Legyen  $\{1, \dots, n\}$  a Peer-ek halmaza.

Ekkor:

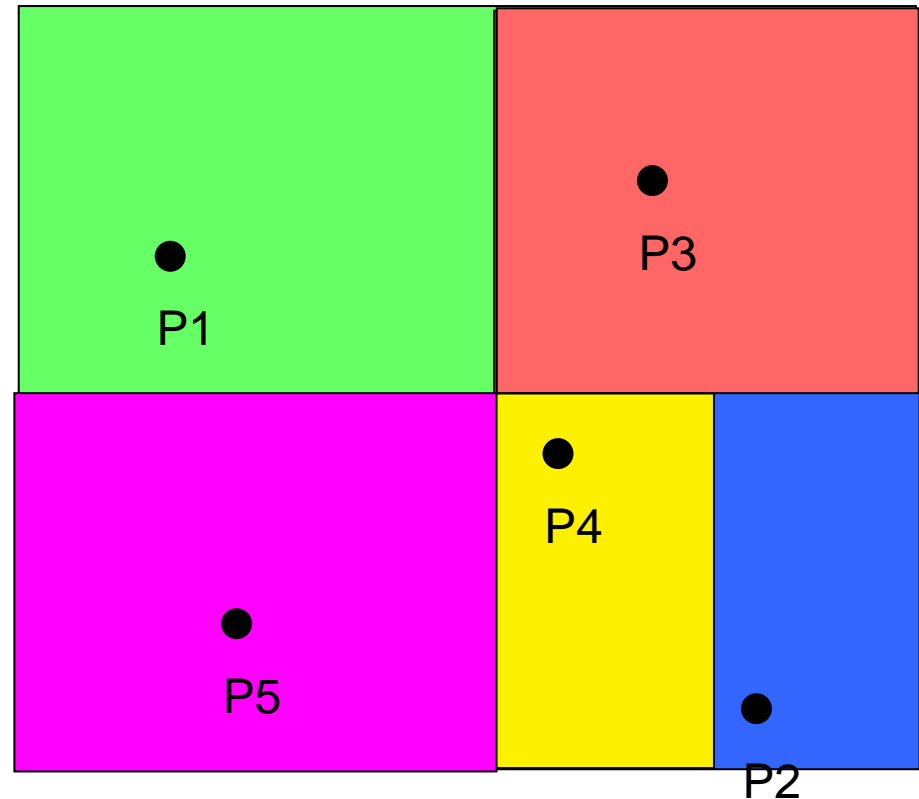
$$\sum_{i=1}^n A(p) = 1$$

Továbbá a szimmetria miatt

$$\forall i \in \{1, \dots, n\} : E[A(i)] = E[A(1)]$$

Így teljesül:

$$1 = \sum_{i=1}^n A(i) = E \left[ \sum_{i=1}^n A(i) \right] = \sum_{i=1}^n E[A(i)] = nE[A(1)]$$



## Egy nem eltalált négyszög

Biz. 2-höz  $P_{R,n} \leq e^{-n \text{Vol}(R)}$

Tekintsünk egy  $R$  négyszöget, melynek területe  $x = \text{Vol}(R)$

Annak a valószínűsége, hogy egy Peer nem  $R$ -be esik

$$1 - x$$

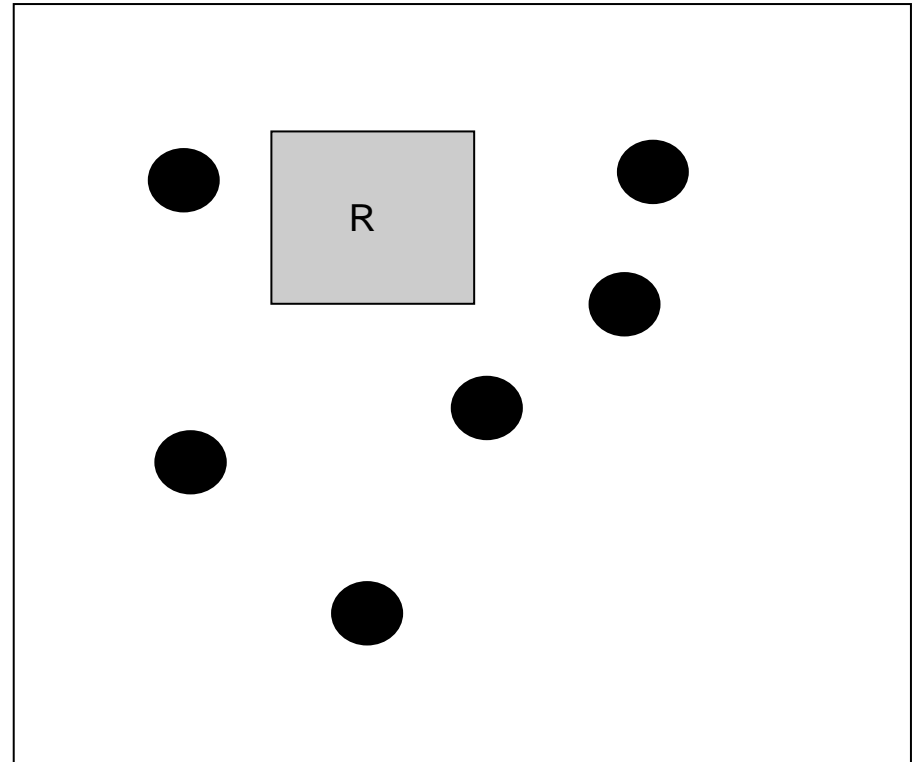
Annak a valószínűsége, hogy mind az  $n$  Peer nem  $R$ -be esik

$$(1 - x)^n$$

Így a valószínűség, hogy egy Peer se esik  $R$ -be legfeljebb

$$(1 - x)^n = \left( (1 - x)^{\frac{1}{x}} \right)^{nx} \leq e^{-nx}$$

Ebből következik a lemma.  $\square$



## Milyen nagy lehet egy nem eltalált négyzög?

Mivel  $P_{R,n} \leq e^{-n \text{Vol}(R)}$ , következik,  
hogyha  $R_i$  egy  $2^i$  területű négyzög, akkor

$$P_{R_i, c2^i \ln n} \leq e^{-c2^i \ln n \text{Vol}(R_i)} = n^{-c}$$

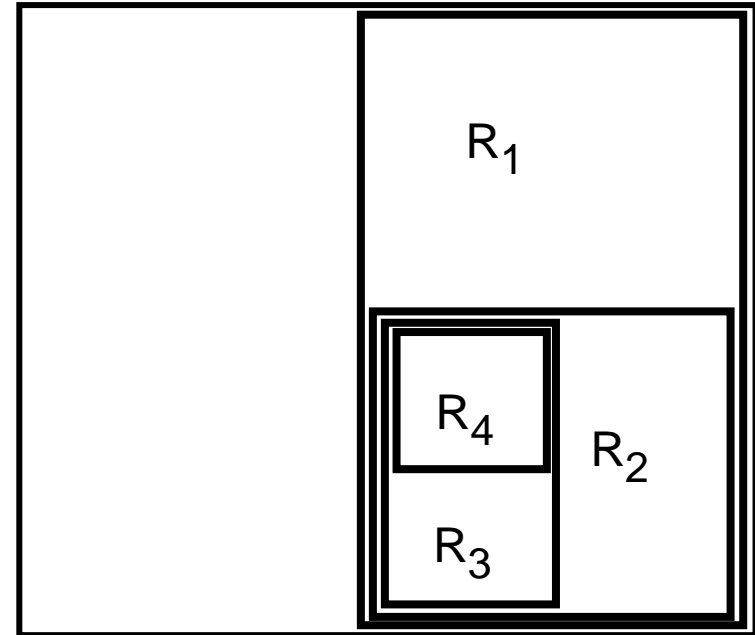
Így annak valószínűsége, hogy  $c2^i \ln n$  Peer  
 $R_i$ -t szétosztja:  $1 - n^{-c}$

Kezdetben a hálózat üres.

Tekintsük  $n$  Peer belépését a hálózatba:

az  $i$ -edik fázisban,  $i \in \{1, \dots, \log \frac{n}{2c \ln n}\}$ ,  
belép  $c2^i \ln n$  Peer.

$$\sum_{i=1}^{\log \frac{n}{2c \ln n}} c \cdot 2^i \ln n = c \cdot \ln n \cdot \sum_{i=1}^{\log \frac{n}{2c \ln n}} 2^i \leq c \cdot \ln n \cdot 2^{\log \frac{n}{c \ln n}} = n$$



## Milyen nagy lehet egy nem eltalált négyszög?

Így annak a valószínűsége, hogy egy  $\frac{2c \ln n}{n}$  területű R négyszög nem lesz szétosztva:  
 $\Pr[ R \text{ valamelyik elődje vagy } R \text{ nem lesz szétosztva} ] \leq n^{-c} \log n$

Legfeljebb  $\frac{n}{2c \ln n}$  ekkora négyszög van, így

$\Pr[ \text{létezik egy ekkora négyszög, ami nem lesz szétosztva} ] \leq \frac{n}{2c \ln n} n^{-c} \log n \leq n^{-c+1}$

**Tétel:** Legyen  $c > 1$  egy konstans.

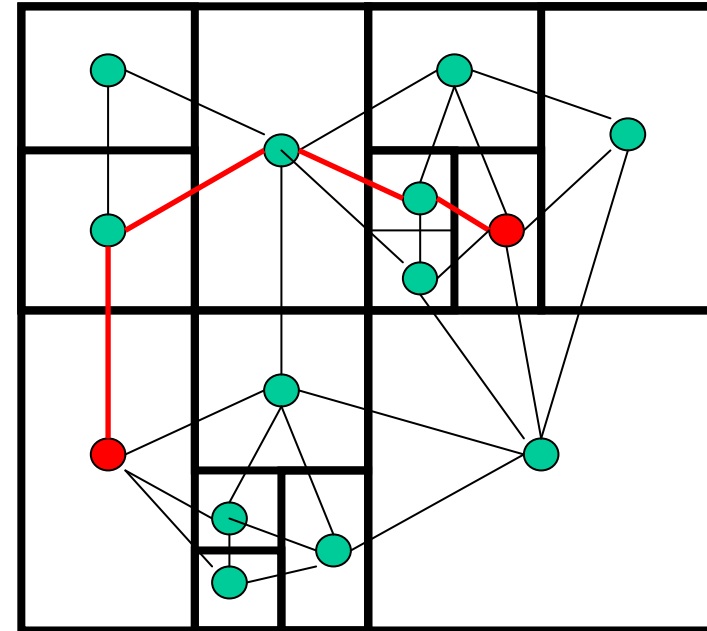
$n$  Peer belépése után nem marad  $\frac{2c \ln n}{n}$  területű  
vagy annál nagyobb négyszög nagy valószínűséggel,  
azaz  $1 - n^{-c'}$  valószínűséggel ( $c' = c - 1$ ).  $\square$

## Milyen kiegyensúlyozottan osztja el az adatokat?

- Ha összesen  $m$  elemet tárolunk, egy adott Peer-en tárolt elemek számának várható értéke arányos a Peer négyszögének területével.
  - akkor minden Peer maximum  $2c (\ln n) m/n$  elemet tárol (nagy valószínűséggel)
  - miközben az átlag  $m/n$  elemet tárol
- Azaz nagy valószínűséggel minden Peer legfeljebb  $2c (\ln n)$ -szer több elemet tárol, mint az átlag.

## Lookup a CAN-ban

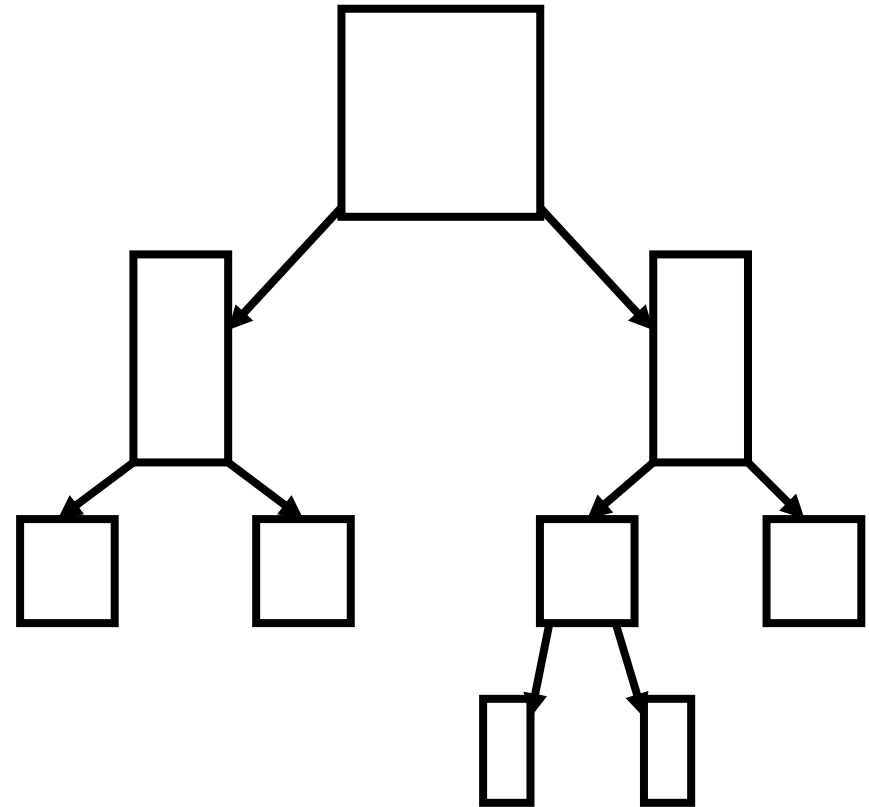
- Először az adat helyét határozzuk meg a hash-függvény értékének kiszámolásával
- A szomszédos négyszögek tulajdonosai között élek vannak
- A kérés az adat helyének irányába továbbítódik
- d dimenziós négyzet/kocka
  - 1 dimenziós: szakasz
  - 2 dimenziós: négyzet
  - 3 dimenziós: kocka
  - 4: ...
- Az élek várható értéke az úton:  $n^{1/d}$
- A csomópontok átlagos fokszáma:  $O(d)$





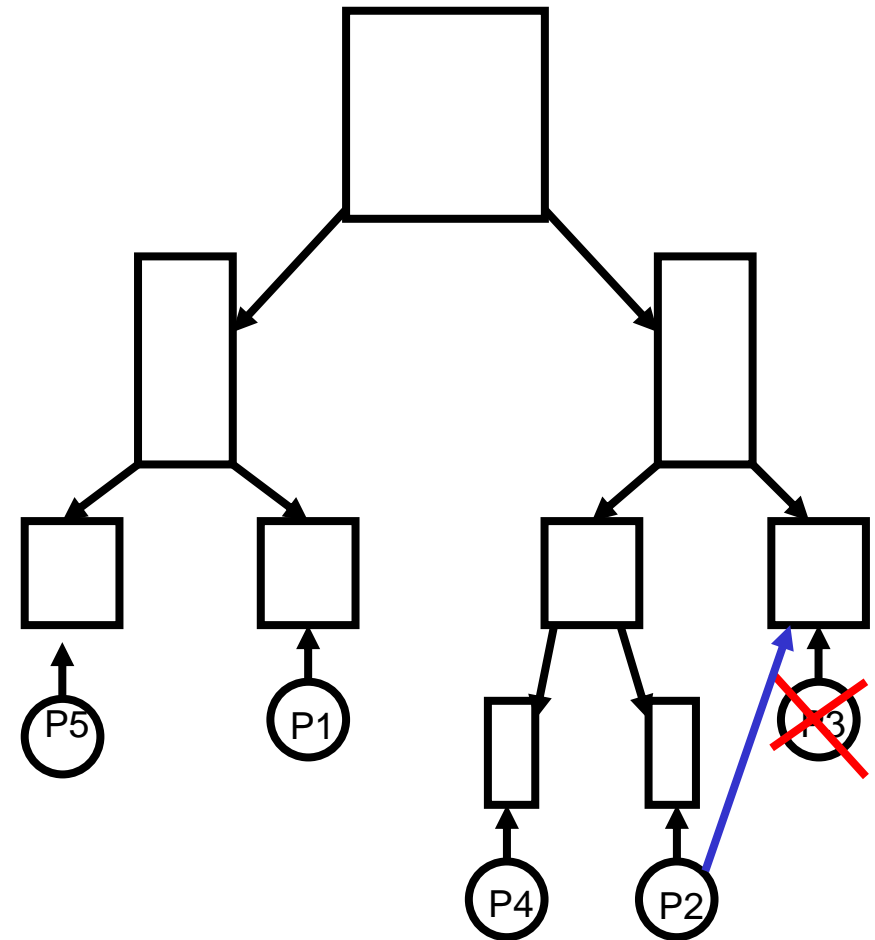
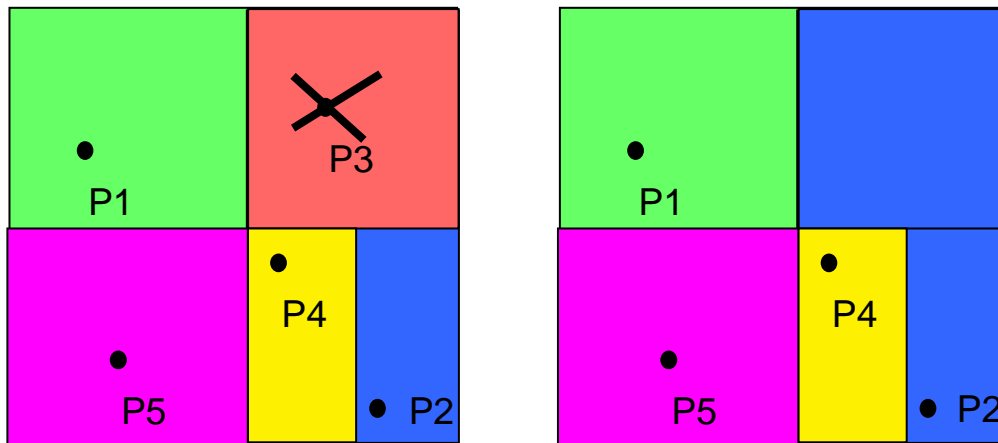
## Befűzés CAN-ba = Véletlen fa

- Véletlen fa
  - Új levelek kerülnek véletlenül befűzésre.
  - Ha az aktuális csomópont belső csomópont, folytassuk véletlenül a bal vagy a jobboldali részében
  - Ha az aktuális csomópont levél, fűzzünk két levelet gyermekként ehhez a csomóponthoz
- A fa mélysége:
  - Várható érték:  $2 \log n + O(1)$
  - Mélység:  $O(\log n)$  nagy valószínűséggel, azaz  $1 - n^{-c}$  valószínűséggel.
- Megfigyelés:
  - A CAN az új Peer-eket úgy fűzi be, mint új leveleket a véletlen fába



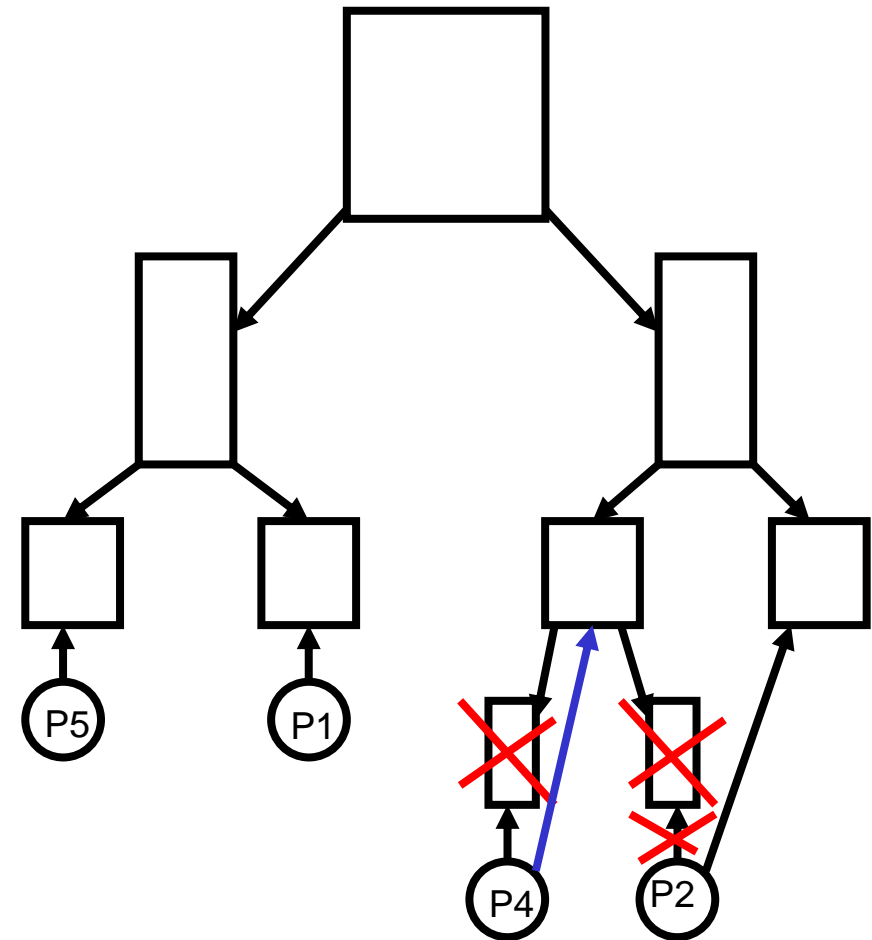
## Peer törlése a CAN-ban

- Mikor egy Peer rendesen kijelentkezik, átadja a zónáját és adatait egy szomszédos Peer-nek.
- Mikor eltűnik, nem biztos, hogy bejelenti előre
  - Ezért a szomszédai rendszeresen tesztelik a jelenlétét
  - Amelyik szomszéd először észleli a Peer eltűnését, átveszi annak a zónáját
- A Peer-ek több tartományt kezelhetnek
- Gyakori befűzés és törlés fragmentáláshoz vezet



## Defragmentálás – Az egyszerű eset

- A fragmentálás megszüntetéséhez időről időre végrehajtunk egy zóna-újrarendelést (zóna=terület)
  - Minden P Peer-hez, amelynek legalább két zónája van,
    - Töröljük P legkisebb zónáját és keressünk ehhez zónához egy új tulajdonos Peer-t
1. Eset:  
A szomszéd a fában nincs kettéosztva
- Mindkét Peer levélnek felel meg a CAN-fában
  - Rendeljük hozzá a zónát a szomszédcsomóponthoz



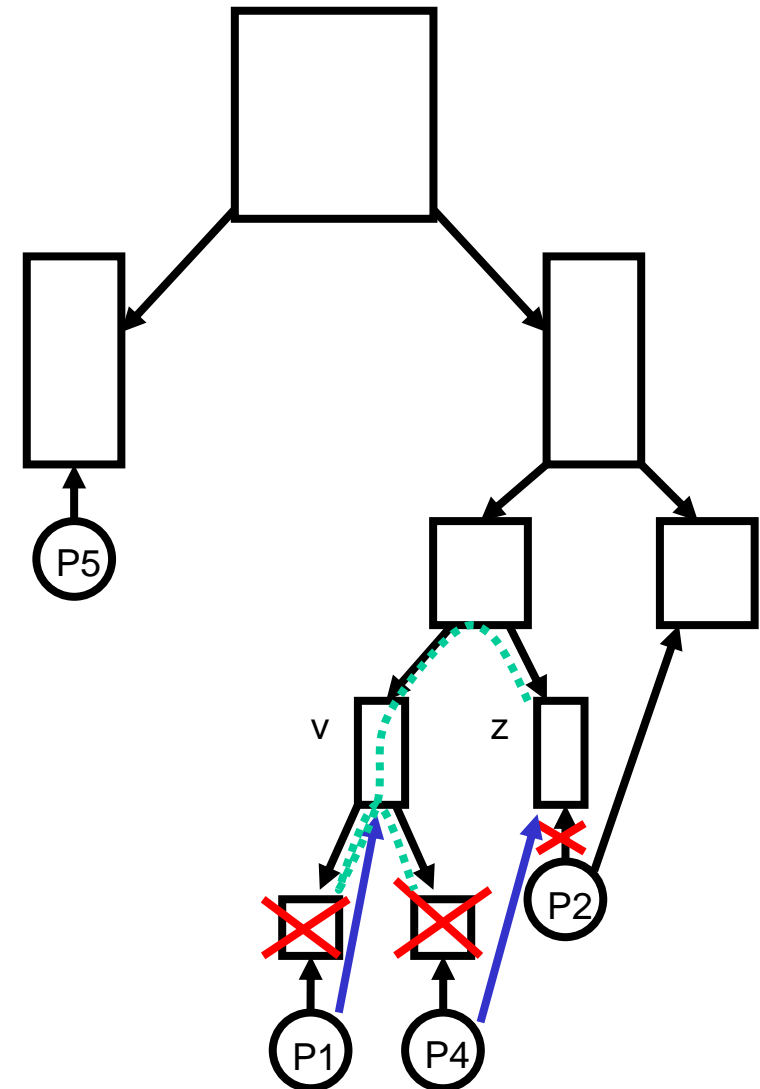
## Defragmentálás – A nehezebb eset

- Minden P Peer-hez, amelynek legalább két zónája van,
  - Töröljük P legkisebb z zónáját és keressünk z-hez egy új tulajdonos Peer-t

### 2. Eset:

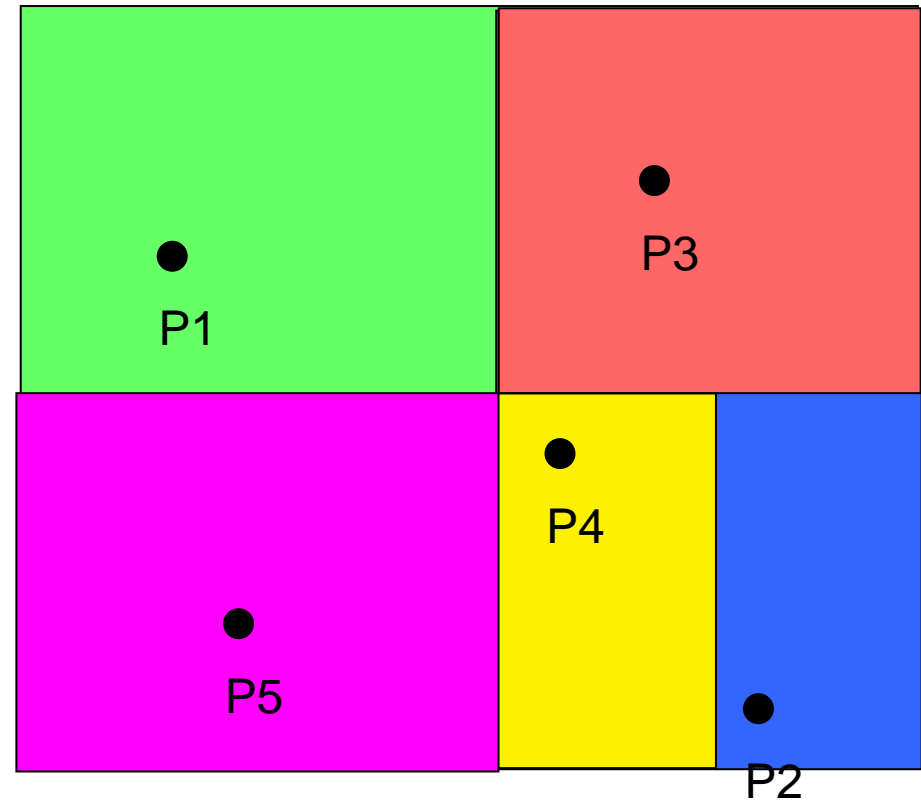
A szomszéd v a fában ketté van osztva

- Hajtsunk végre a fában v-ből egy mélységi keresést addig, amíg két szomszédos levelet találunk
- Rendeljük hozzá az egyik levél Peer-jéhez mindkét levél zónáját és
- Válasszuk a másik levél Peer-jét a z zóna tulajdonosának



# CAN Értékelése

- Előny
  - Egyszerű robusztus viselkedés
  - Kiegyensúlyozza az adatok eloszlását a Peer-ek között
  - Alacsony fokszám
  - A hálózat többszörösen összefüggő, ezáltal robusztus
  - Különböző utakat ismer a célhoz, ezáltal tud utakat optimalizálni
- Hátrány
  - Átmérő konstans dimenzió esetén polinomiális:  $d$  dimenzió esetén  $O(n^{1/d})$

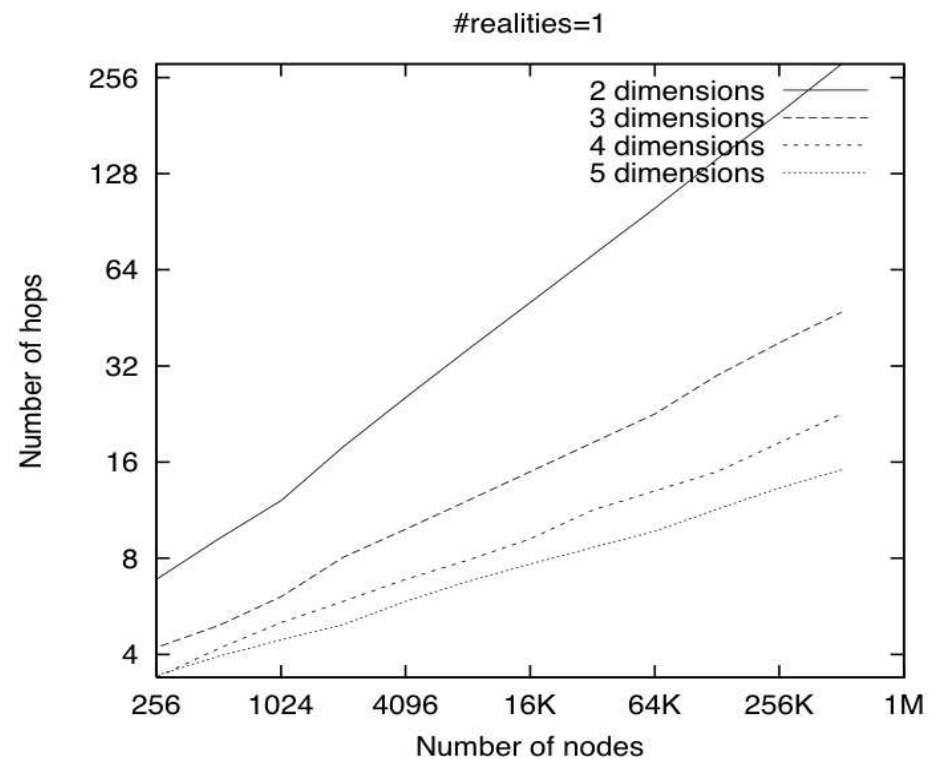


## Rendszerjavítások a CAN-hoz

1. Többdimenziós terek
2. Különböző valóságok
3. Távolságmérika a routinghoz
4. Zónák túlpakolása
5. Többszörös hashing
6. Topologia-függő hálózatkonstrukció
7. Egyenletesebb partícionálás
8. Caching, Replikáció és und Hot-Spot-Management

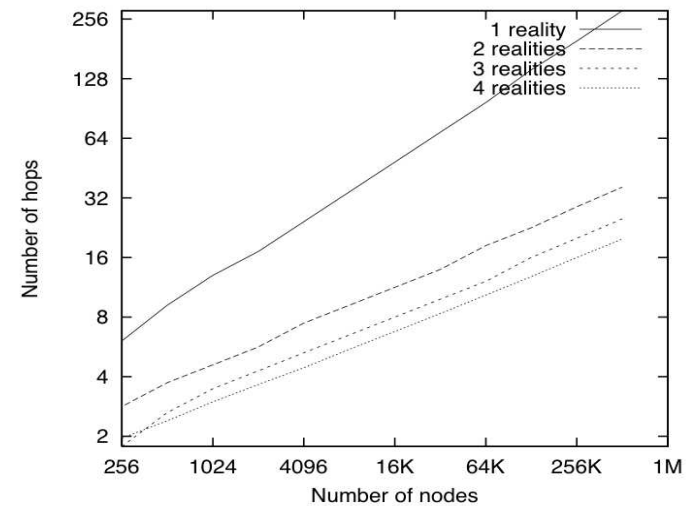
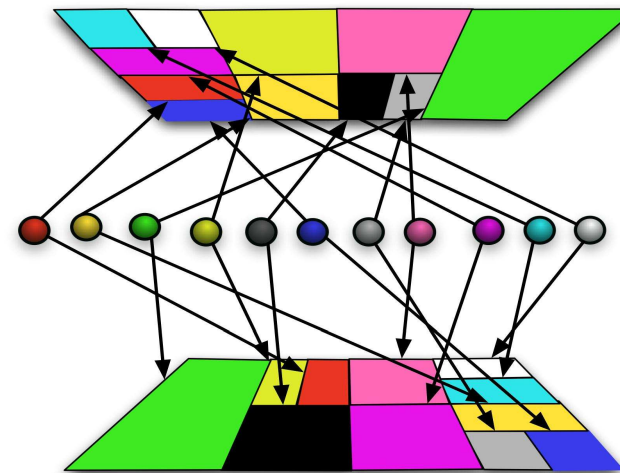
# Többdimenziós terek

- d-dimenziós tér (2-D helyett)
  - 1: szakasz
  - 2: négyzet
  - 3: kocka
  - ...
- A várható útvonalhossz d-dimenzió esetén  $O(n^{1/d})$
- A szomszédok számának várható értéke  $O(2^d)$



# Több valóság

- Szimultán  $r$  CAN-hálózatot építünk fel
- Minden CAN-hálózatot valóságnak nevezünk
- Ha egy adatot keresünk,
  - ugorhatunk a valóságok között
  - Azt a valóságot választjuk, amelyben a távolság a célhoz minimális
- Előny
  - Robusztus
  - Rövidebb utak

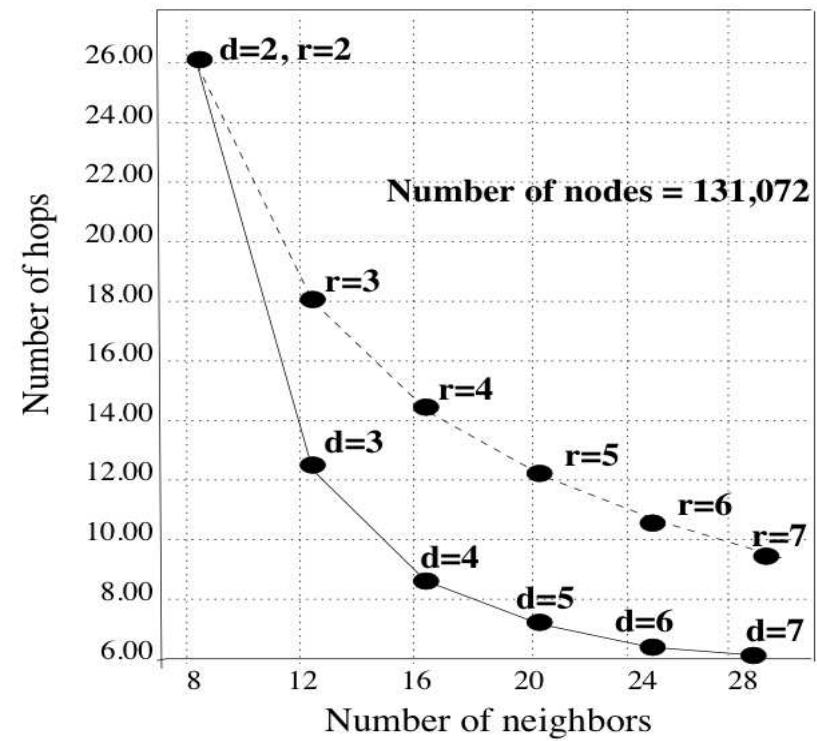




# Több realitás vagy több dimenzió

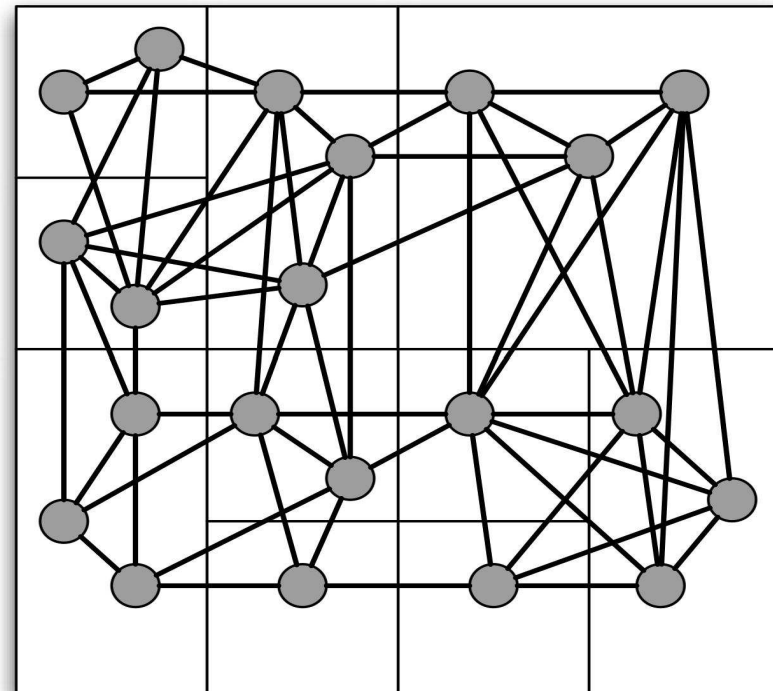
- Több dimenzió jobban csökkenti az utak hosszát
- Több realitás robusztusabb hálózatot eredményez

- ————— increasing dimensions, #realities=2
- - - - - - increasing realities, #dimensions=2



## Zónák túlpakolása

- Minden zónában MAXPEERS (z.B. 10) Peer lehet
  - Minden Peer ismer minden Peer-t a saját zónájában
  - és mindegyik ismer egy Peer-t a szomszéd zónában
  - Ezáltal az utak nem lesznek hosszabbak
- Az utak  $O(\text{MAXPEERS})$ -szer rövidebbek lesznek
- A várakozási idő megrövidül, ha
  - minden Peer a szomszéd zóna földrajzilag legközelebbi Peer-jét választja
- Jobb hibatolerancia



## Távolságmétrikák a routinghoz

- Az RTT (round trip time) mérése által becsüljük a távolságot
- Ezen metrika szerinti legközelebbi szomszédokat részesítjük előnyben
- Előny:
  - Csökken a várakozási idő
- Topológiafüggetlen hálózatkonstrukcióval több időmegtakarítás érhető el

# Többszörös Hashing

- Az adatokat nem csak egyszer, hanem több helyen is tároljuk,
  - úgy, hogy a kulcsot a  $k \in \{1,2,\dots, \text{COPIES}\}$  számmal kombináljuk
- A robusztusság ezáltal növekszik
- Kisebb távolságok
  - A legközelebbi másolatot keressük
  - Az routing-útvonalak hossza indirekt összefügg a másolatok számával.  
(pl. 1-dimenzióban fordítottan arányos a másolatok számával)

## Topológiafüggő hálózatkonstrukció

- A mért várakozási idők  $m$  kitüntetett Peer-hez – amiket **landmark**-oknak nevezünk – információként szolgálnak a pozícióról
- A várakozási időket sorbarendezzük
- A sorbarendezt lista kulcsként szolgál:  $m!$  lehetséges kulcs
- Ezt kulcsot képezzük le
  - A leképezéshez nem egy hash-függvényt használunk
  - Minden permutációnak megfeleltetjük a koordináta-rendszer egy tartományát, és a Peert ezen a tartományon képezzük le egy véletlen pontra.
- Ezáltal
  - Közeli csomópontok azonos (közeli) tartományba kerülnek
  - A várakozási idők drasztikusan csökkennek (közeli tartományok esetén)
- De
  - A landmark-csomópontok választása nehéz
  - Fennál a veszélye a feladatok egyenletlen eloszlásának

## Irodalom

- Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker: **A scalable content-addressable network**. In *Proc. ACM SIGCOMM*, 161-172, 2001.
- M. Jovanovic, F. Annexstein, and K. Berman: **Scalability Issues in Large Peer-to-peer networks: A case study of Gnutella**.
- Napster: <http://computer.howstuffworks.com/file-sharing.htm>