Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba
TMIT
2018

Strukturált P2P

P2P Keresés

- Strukturálatlan P2P rendszerek
  - Gnutella, KaZaa, stb…
  - Véletlenszerű keresés
    - Nincs információ a fájl lehetséges tárolási helyéről
  - Nagy terhelés a rendszeren
  - Minél több helyen tárolva, annál kisebb a keresés által generált terhelés

P2P Keresés (II)

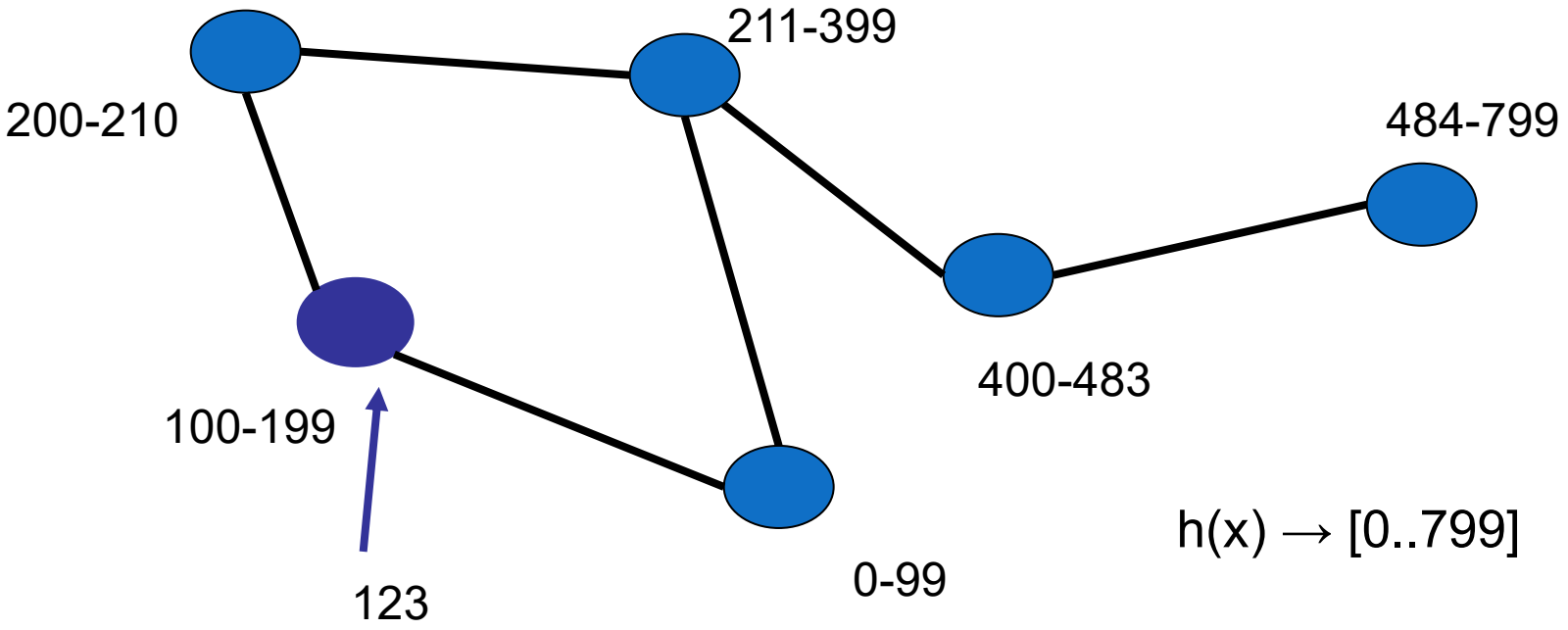
- Strukturált P2P rendszerek
  - Irányított keresés
    - Kijelölt peer-ek kijelölt fájlokat (vagy azok fele mutató pointereket) kell tároljanak
      - Mint egy információs pult
    - Ha valaki keresi a fájlt, tudja kit kérdezzen

- Elvárt tulajdonságok
  - Elosztottság
    - Felelősség elosztása a résztvevők között
  - Alkalmazkodás
    - A peer-ek ki és bekapcsolódnak a rendszerbe
    - Az új peer-eknek feladatokat osztani
    - A kilépő peer-ek feladatai újraosztani

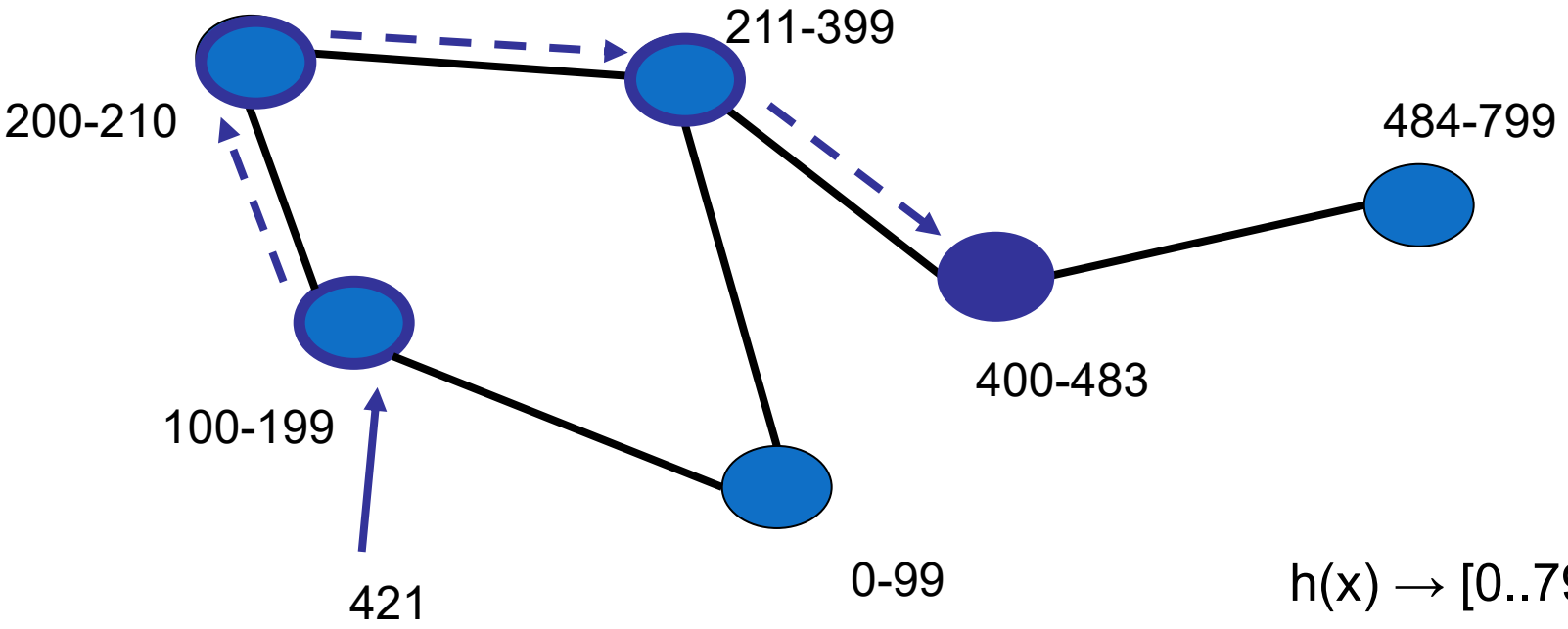
Elosztott hash táblák - DHT

- Distributed Hash Tables
- Egy hash függvény a keresett fájlhoz egy egyéni azonosítót társít
  - Példa: h(„P2P_előadás”) -> 123
- A hash függvény értékkészletét elosztja a peer-ek között
- Egy peer-nek tudnia kell minden olyan fájlról, melynek a
- hash-elt értéke a saját értékkészletébe tartozik
  - Vagy saját maga tárolja a fájlt….
  - Vagy tudja ki tárolja a fájlt.

DHT példa



DHT példa



h(x) → [0..799]

DHT útválasztás

- Minden peer mely információt tárol egy fájlról elérhető
- kell legyen egy „rövid” úton
  - Korlátozott számú lépésen belül

- Korlátozott számú szomszéd
  - Skálázható a rendszer méretétől függetlenül

- A különböző DHT megoldások lényégében az
- útválasztásban térnek el
  - CAN, Chord, Pastry, Tapestry, Kademlia

CAN

- https://www.tribler.org/Khashmir/

Kademlia

Kademlia: A peer-to-peer information system based on the XOR metric

- Petar Maymounkov and David Mazieres
- New York University

- *Proceedings of the 1st International Workshop on Peer Systems (IPTPS '02)*, pages 53-65, March 2002.

- Kademlia does not have a main implementation called Kademlia,

instead the Kademlia DHT has been implemeted in different languages

such as C (KadC), Python (Kashmir), and Java (Plan-x)

Kademlia

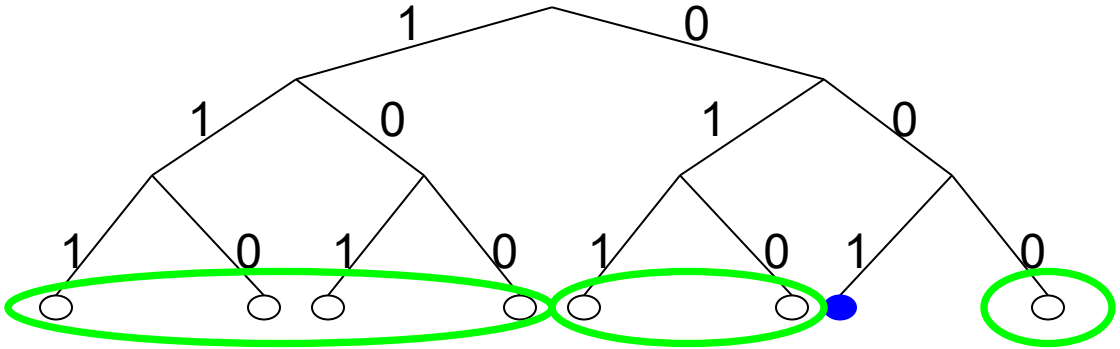
- P2P adattároló hálózat
  - Kulcs-érték párok tárolása
    - egyedi, 160 bites kulcsok
  - DHT
- Kialakítandó környezet (aka specifikáció)
  - Bármelyik node bármikor eltűnhessen
  - A node-ok legyenek egyenletesen terheltek
    - tárolási kapacitás
    - sávszélesség
- Cél
  - Gyors adatelérés
  - Minimális számú vezérlő üzenet

Kademlia

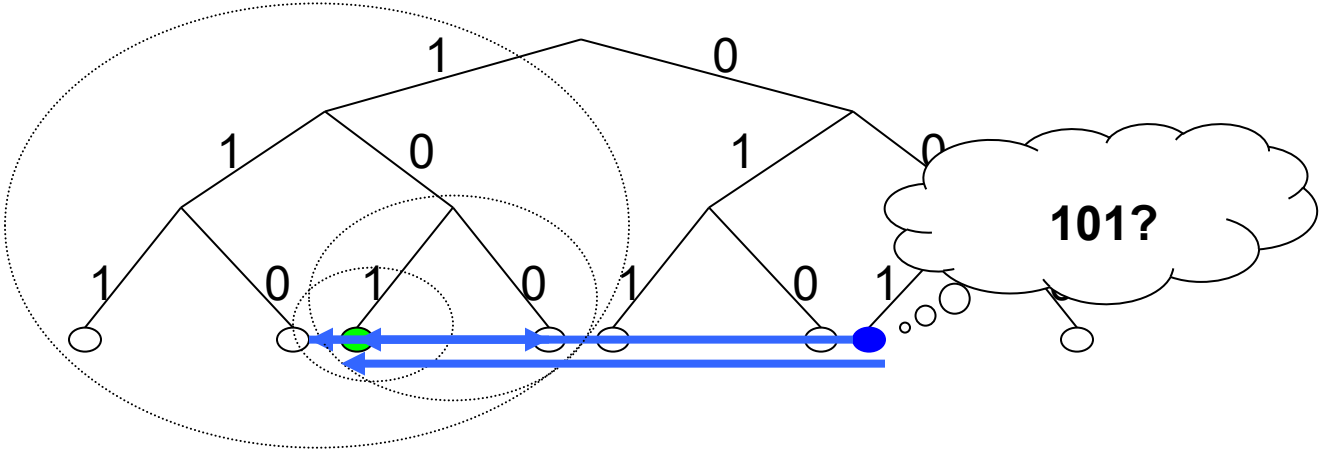
- A routing-tábla rugalmas
  - Közelségi alapú (proximity-based) routing
  - Laza követelmények
    - minimális fenntartás
- Bejövő és kimenő üzenetek eloszlása azonos
  - A hálózat önmegerősítő
- Csak *log n* node elérését kell ismerni

Routing tábla

- Rugalmas kialakítás
  - Mindegy, hogy kit ismerek az adott tartományban
- A kimenő és bejövő eloszlás azonos



Kademlia – routing



- Minden hop eggyel kisebb ágra visz a cél körül
  - Az adott ágon belül bármelyik node-hoz ugorhatunk

XOR - topológia

- Definíció: d(x,y) = x XOR y
  - Szimmetrikus: d(x,y) = d (y,x). Kevesebb vezérlő üzenet
  - Egyértelmű: minden $x \in \{0,1\}^{160}$ és $l \in N$-hez pontosan egy olyan $y \in \{0,1\}^{160}$ létezik, hogy d(x,y) = l.

- A nagyobb helyiértékű bitek eltérése nagyobb távolságot
- eredményez:
  - x = 010101
  - y = 110001
  - x XOR y = 100100, azaz 32 + 4 = 36

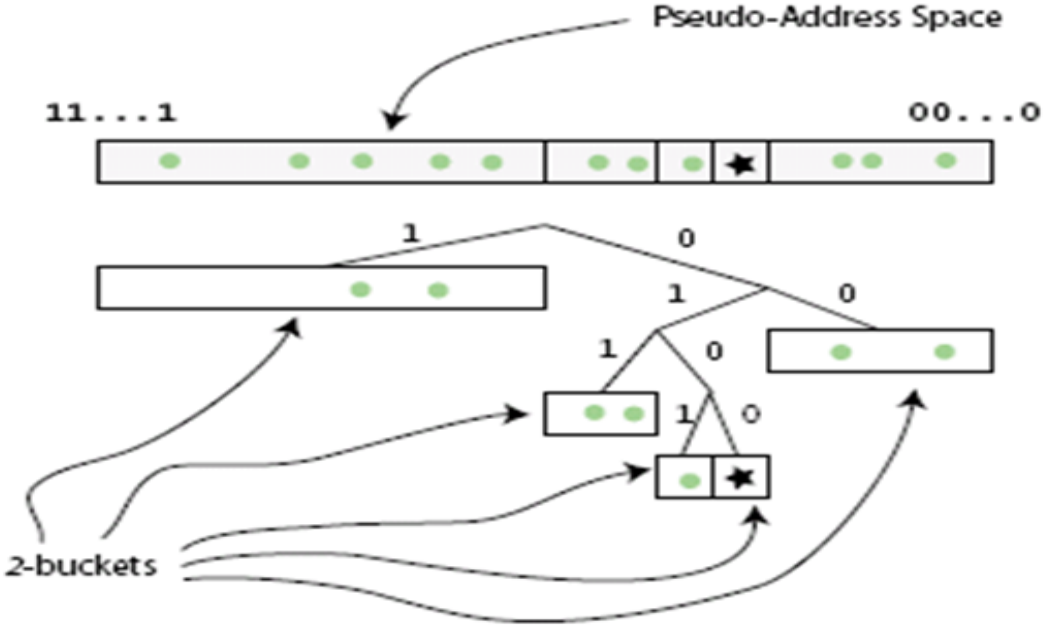
- Geometriai értelmezés:
  - A közös ágon lévő csomópont közelebb van, mint a többi

Adatstruktúrák

- Kapcsolatok
  - (nodeID, IP:UDP_port) párok
- k-vödör
  - tárolóegység maximum k db. kapcsolathoz
    - alapértelmezésben k=20
    - Egy A állományra (ID_A) vonatkozó infókat a (XOR metrika szerint) hozzá legközelebb álló k db. csomóponton fogunk tárolni
      - Annak a valószínűsége, hogy mind a k csomópont egyszerre kilépjen kellően kicsi legyen
- Routing tábla
  - Műveletek: kapcsolat létesítése, törlése
  - Egy k-vödrökből álló fa
    - Minden vödör a fa egy tartományáért felelős

Útválasztó tábla

- A 0100 csomópont útválasztó táblája



Keresés

- Cél: A T célhoz legközelebbi k node megtalálása, ahol $T \in \{0, 1\}^{160}$
- $find\_node_n(T)$: visszaadja azt a k darab címet az n csomópont
- útválasztó táblájából, amelyik T-hez legközelebb áll
  - Egy vagy több k-vödörből
  - Ha nincs k node az összes vödrében, akkor kevesebbet ad vissza
  - A válaszok minden lépésben visszamennek a keresést indító peer-hez
- Keresés:
  - n_0: a keresést végző node
  - $N_0 := \{\emptyset\}$
  - $N_1 = min_k\{find\_node_{n_0}(T), N_0\}$
  - $N_2 = min_k\{find\_node_{n_1}(T), N_1\}$
  - …
  - $N_i = min_k\{find\_node_{n_{i-1}}(T), N_{i-1}\}$
    - Ahol n_i $(i > 0)$ az N_i halmaz tetszőleges címe
- A keresés akkor ér véget, ha N_i csak olyan címeket tartalmaz, amelyeket n_0
- már megkérdezett
  - Az egymás utáni $find\_node_n(T)$ hívások egyre kisebb tartományhoz tartozó k-vödörből adnak választ

Párhuzamos keresés

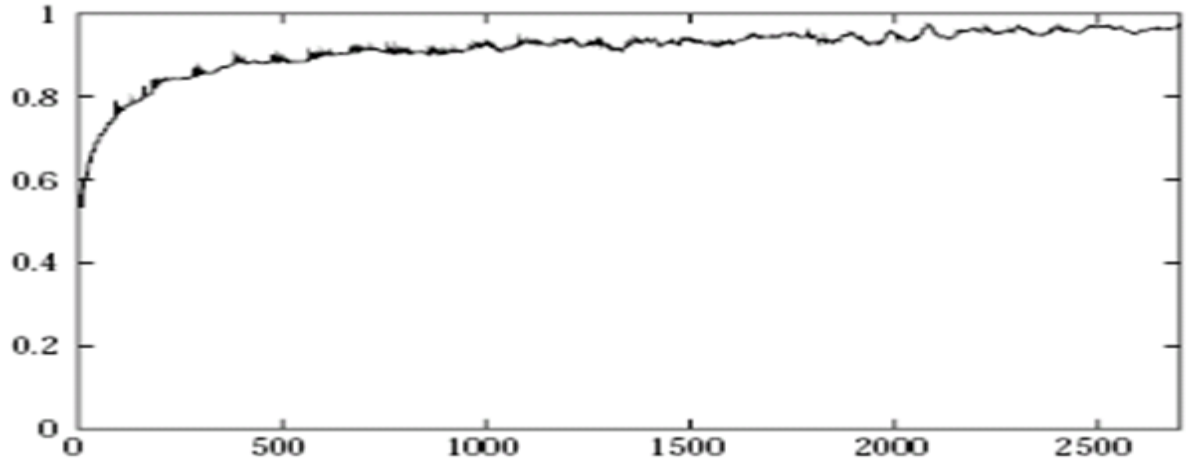
- Nagyobb forgalomért cserébe gyorsabb keresés
- Cél
  - A keresés menjen a gyorsabb / közelebbi gépeken keresztül
  - Kerüljük el a kilépett node-ok miatti timeout-okat
- Megoldás:
  - egyszerre $\alpha > 1$ párhuzamos $find\_node_n(T)$

Kapcsolatok fenntartása

- Ha egy node nem válaszol, töröljük a routing táblából
- Minden egyes node amelyiktől „hívást” kapunk bekerül a routing
- táblánkba ha…
  - Megnézzük, hogy a k-vödör ahova kerülne tele van-e
    - Ha nem, betesszük
    - Ha igen, megpingeljük azt a node-ot, melyről legrégebb hallottunk
      - Ha válaszol, az új node-ot nem tesszük be a táblába
        - Ha a régi él, akkor valószínű hogy később is élni fog, kár elveszteni
        - Mérések alapján a Gnutella hálón
        - DoS támadások ellen is véd
      - Ha nem válaszol, töröljük, és felveszi az új node-ot

- Az XOR topológia szimmetriája miatt a minket hívó node-ok eloszlása éppen
- a routing táblánkéval egyező eloszlású lesz
  - Formálisan: annak valószínűsége, hogy egy $[2^i, 2^{i+1}]$ távolságra levő node-tól kapunk üzenetet konstans, azaz i-től független

Node-ok elérhetősége a Gnutella hálón



- x-tengely: az elérhetőség időtartama percben
- y-tengely: azon node-oknak, amelyek x percen át elérhetőek, mekkora hányada lesz x+60 perc múlva is elérhető

Érkezés, távozás, frissítés

- Node érkezése:
  - Egy már bentlévő node-tól néhány címet elkér
  - ‘Megkeresi’ önmagát
    - Megkapja a saját ID-jéhez legközelebb álló k node címét
    - Ezektől a node-októl megkapja a tárolandó infókat
  - Bekerül mások k-vödreibe, elkezdi feltölteni saját k-vödreit
- Node távozása
  - Nincs semmi tennivaló
- Periodikus frissítés
  - Ha egy csomópont tárol egy értéket, periódikus időközönként megkeresi a k-1 másik csomópontot amelyik legközelebb áll az adott értékhez
  - Tárolja az értéket ezeken a csomópontokon

Alkalmazások

- A Kademliát használó néhány alkalmazás:
  - Overnet háló: Overnet, eDonkey, mlDonkey
    - 2006-ban 645.000 párhuzamos felhasználó
    - A RIAA betiltatta 2006-ban
    - Az eredeti eD2k hálót helyettesítette
  - Kad háló: eMule, mlDonkey, aMule
    - eMule fejlesztői az Overnet helyett saját Kademlia implementációt készítettek
  - Directconnect: RevConnect
  - Trackerless nt: A em

Tapestry

Tapestry: A Resilient Global-scale Overlay for Service Deployment

- Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz

- *IEEE Journal on Selected Areas in Communications, January 2004, Vol. 22, No. 1.*

Tapestry

- Egy *elosztott, hibatűrő, adaptív* lokalizáló és útválasztó infrastruktúra
- Utótag (*suffix*) alapú útválasztás (Kademliától eltérő metrika)
- A Plaxton algoritmus ötletére alapoz
  - C.G. Plaxton, R. Rajaraman and A.W. Richa, *Accessing Nearby Copies of Replicated Objects in a Distributed Environment*., 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '97), pp 311-320, Newport, RI, USA, 1997 http://citeseer.ist.psu.edu/plaxton97accessing.html

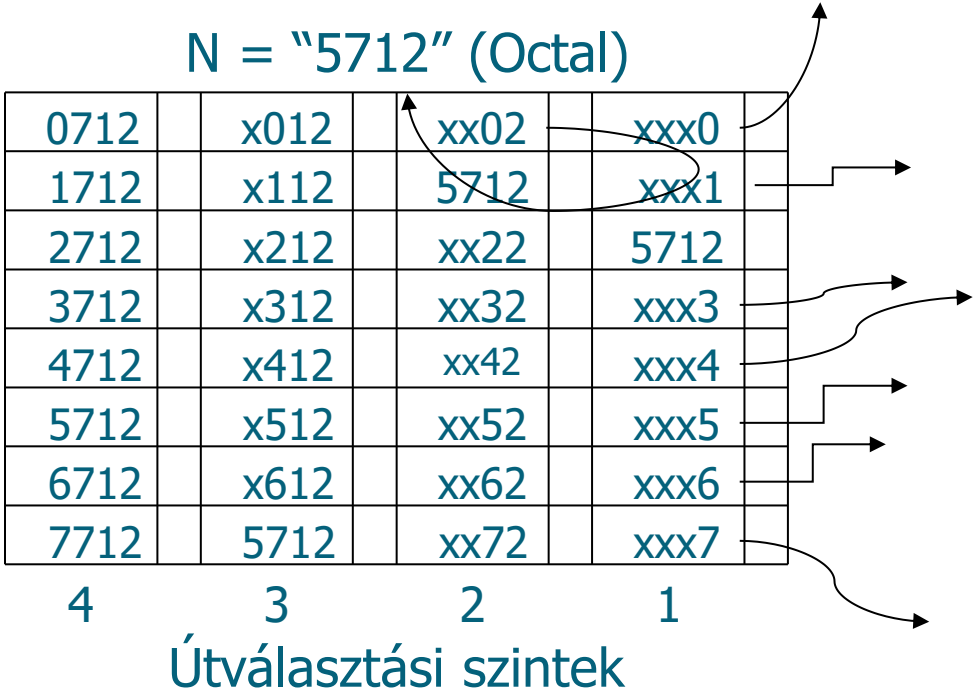
Plaxton/Tapestry címzés

- Bármely csomópont lehet:
  - Szerver – állományokat tárol
  - Router – csomagokat továbbít
  - Kliens – kéréseket kezdeményez
- Név (cím-) tartomány
  - Csomópontok és állományok egyaránt
  - Megfelelően nagy az ütközések elkerüléséhez
    - 160 bit, 40 hexa számjegy, $16^{40}=2^{160}$ cím
- ~ Kiegyensúlyozott eloszlás a tartományon belül
  - Hash algoritmus

Neighbour Map

- Legyen N egy csomópont (IP cím, ID)
  - utótag(N, k) = az utolsó k számjegy az ID-ből
- Minden csomópontban *szomszédossági térkép*
- (neighbour map)
  - Annyi szint, ahány számjegy az ID-ben
  - Minden szinten annyi bejegyzés, ahányas számrendszerben címzünk
  - A j. szint (j-1) hosszúságú utótagoknak felel meg
  - Az i. bejegyzés a j. szinten – a fizikailag legközelebb álló olyan csomópont IP címe, mely ID-je [„i” + utótag(N, j-1)]-re végződik
    - Példa: a 2. bejegyzés a 5712 csomópont térképének 3. szintjén az a 212-re végződő ID-jű csomópont IP címe, mely fizikailag legközelebb áll az 5712 ID-jű ponthoz

Neighbour Map



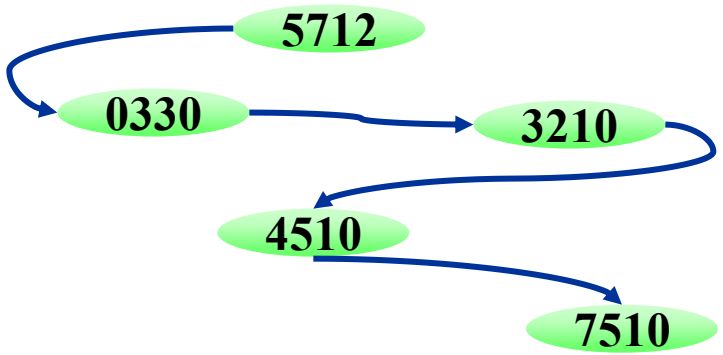
Utótag alapú útválasztás

- Pontról pontra való továbbítás, számjegyenként
  - **** → ***0 → **10 → *510 → 7510
- Hasonlít a *longest prefix match* alapú IP útválasztásra
- Mindegy melyik irányból közelítünk
  - Az eredeti Tapestry javaslat – utótag alapú
  - Jelenleg – előtag alapú

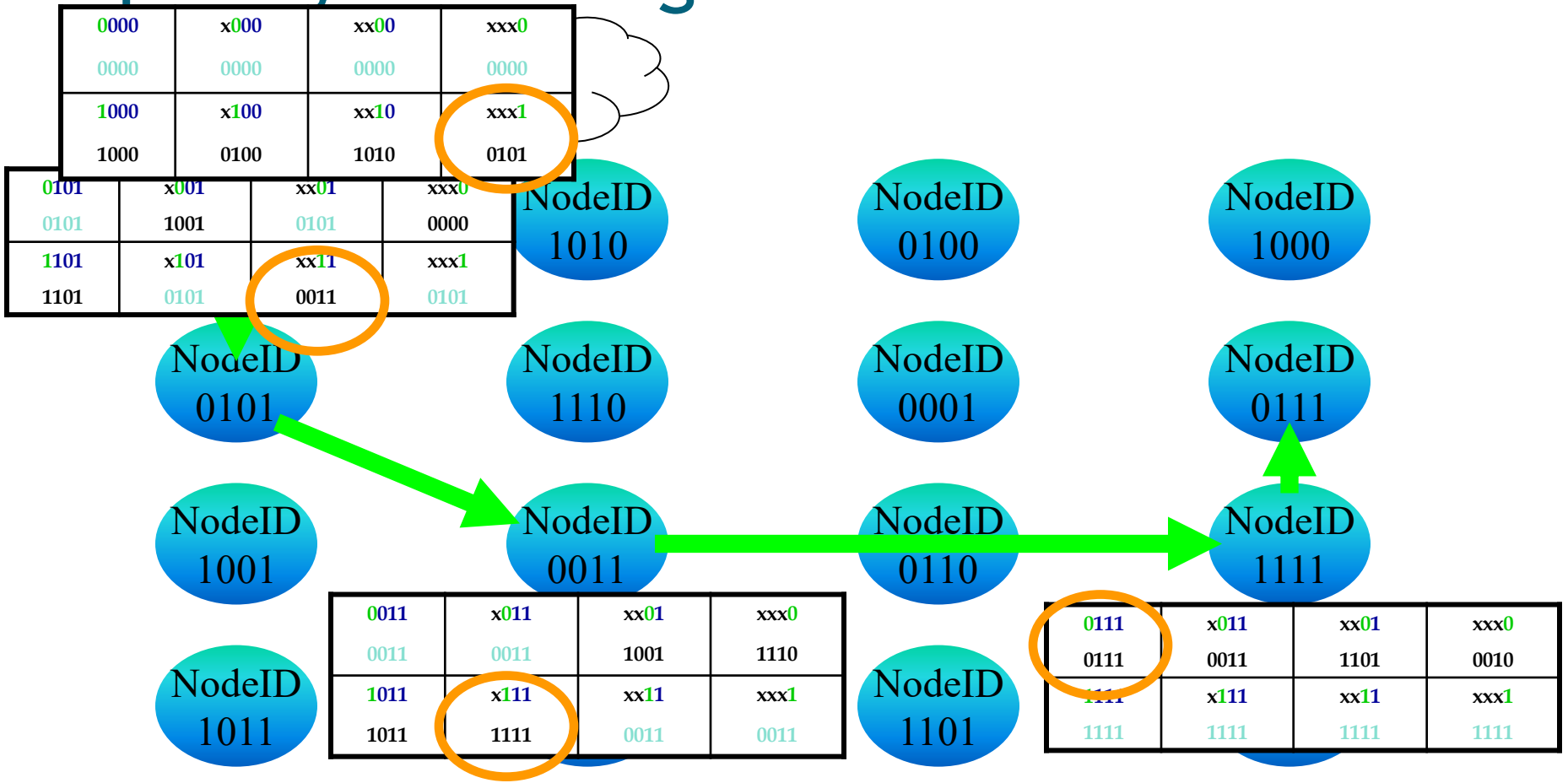
Példa

5712 → 7510

5712	0	1	2	3	4	5	6	7
0330	0	1	2	3	4	5	6	7
3210	0	1	2	3	4	5	6	7
4510	0	1	2	3	4	5	6	7
7510	0	1	2	3	4	5	6	7



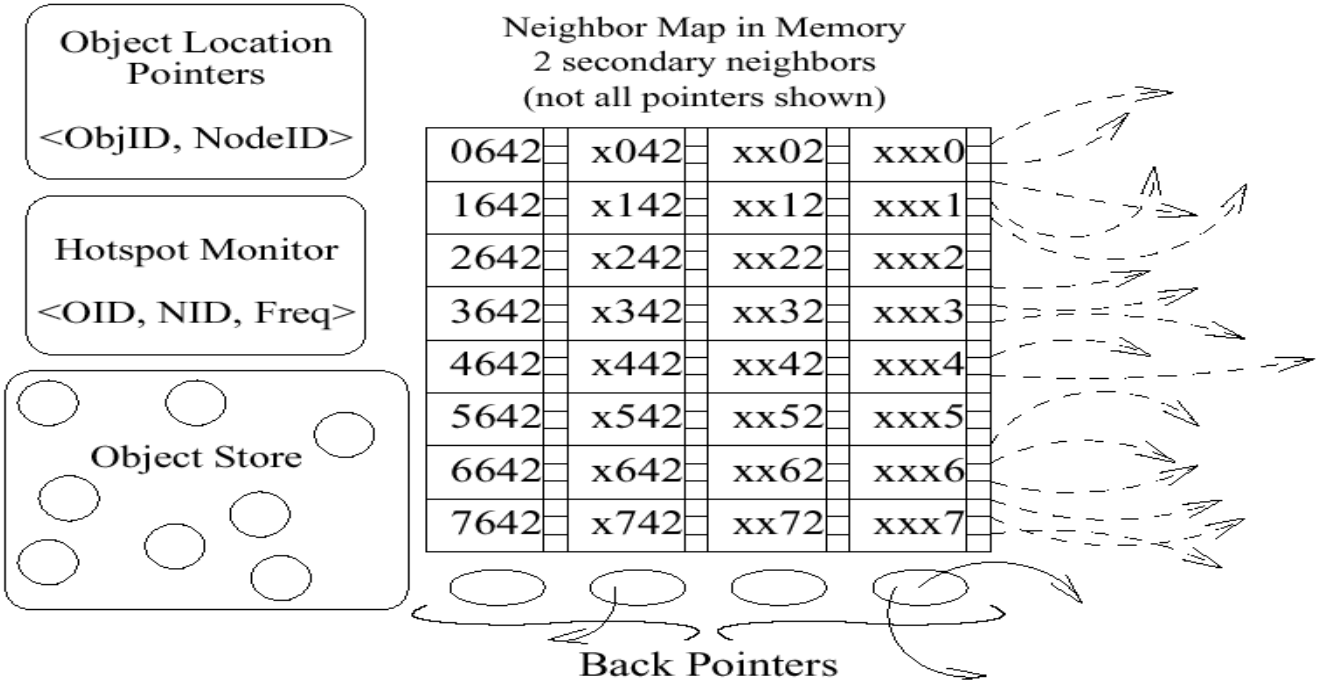
Tapestry – routing



Tapestry csomópont N

- Neighbour Map
- Object Store
  - A helyileg tárolt állományok
- Object Location Pointers
  - Információk bizonyos állományok tárolási helyéről
  - <ObjectID, NodeID>
- Back Pointers
  - Azokra a pontokra mutatnak, melyek szomszéduknak tekintik N-t
- Hotspot Monitor
  - <ObjectID, NodeID, Frequency> - segítenek a cache kezelésében

Tapestry csomópont



Root Node (Plaxton)

- Adott egy A állomány (ID_A)

- Az A állomány gyökere (root node) az az R csomópont, melyre igaz a következő:

$$utótag(ID_A, k) = utótag(ID_R, k)$$

  - és nincs olyan más csomópont X melyre igaz lenne, hogy

$$utótag(ID_A, k+1) = utótag(ID_X, k+1)$$

- Ha több ilyen pont van, a legnagyobb címmel rendelkező lesz a root node

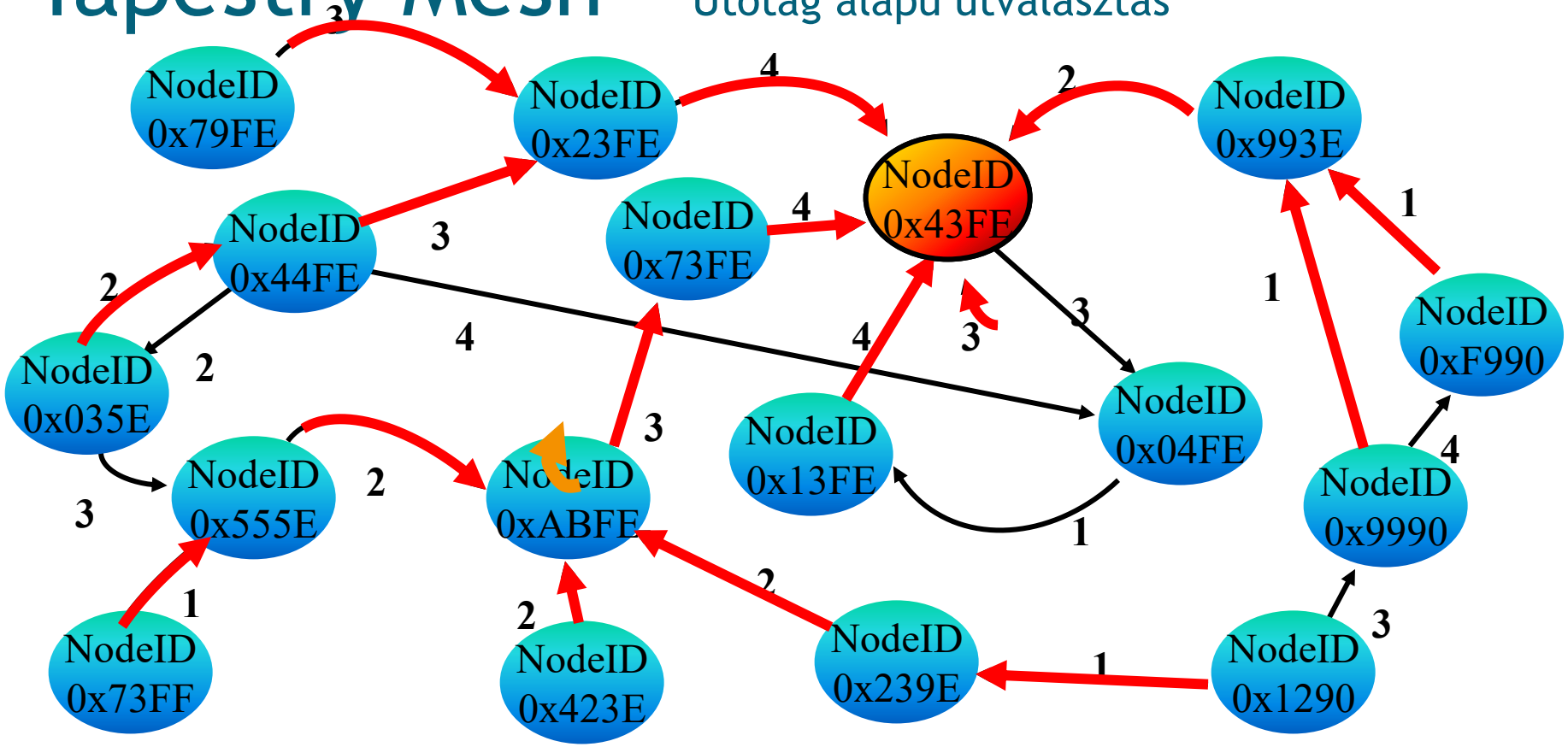
A feszítőfa

- Root(A) az a pont, ahova mindeki fordul ha A-ra kíváncsi
  - Minden A állományhoz egy Root(A) gyökerű feszítőfa tartozik

- A hálózat bármely pontjáról, véges számú lépés alatt eljutunk a
- feszítőfa gyökeréhez
  - Számjegyenként egyre közelebb kerülünk, ameddig egy üres szomszéd bejegyzéshez érünk
  - Egy utolsó ugrásként egy shortcut vezet a root-hoz
  - Információt szerzünk az A állományról

- Statikus megoldás, a hálózat teljes ismerete szükséges
  - Az összes shortcut-ot előre ki kell számolni

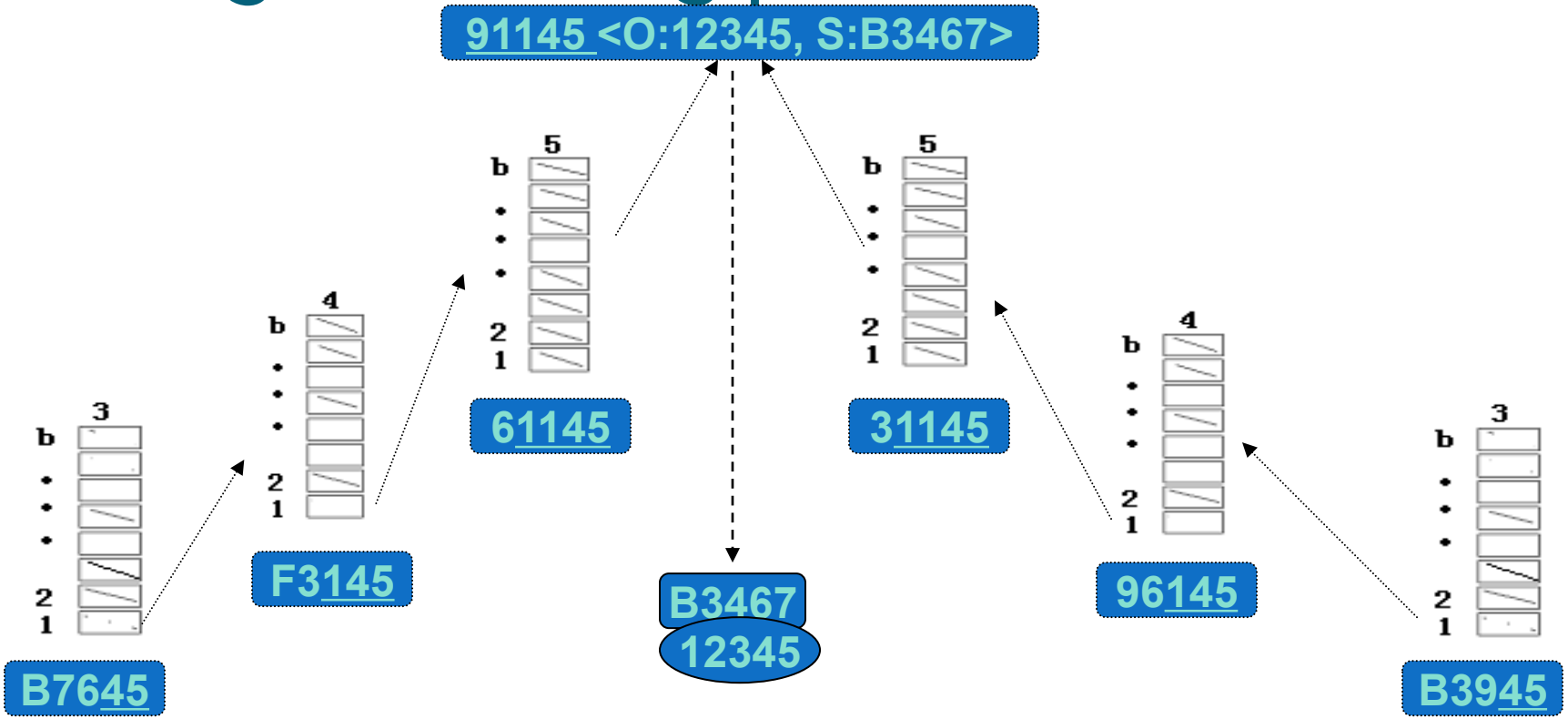
Tapestry Mesh - Utótag alapú útválasztás



Root node (Tapestry)

- Surrogate (helyettesítő) routing
  - Elosztott megoldás a root kiszámolására
  - Ha egy üres szomszéd bejegyzésre bukkan, kiválasztja az azon a szinten levő következő nem üres bejegyzést
  - Ha egy szint után egyetlen bejegyzés sincs saját magán kívül, megáll
  - Ez a pont lesz a surrogate (root)

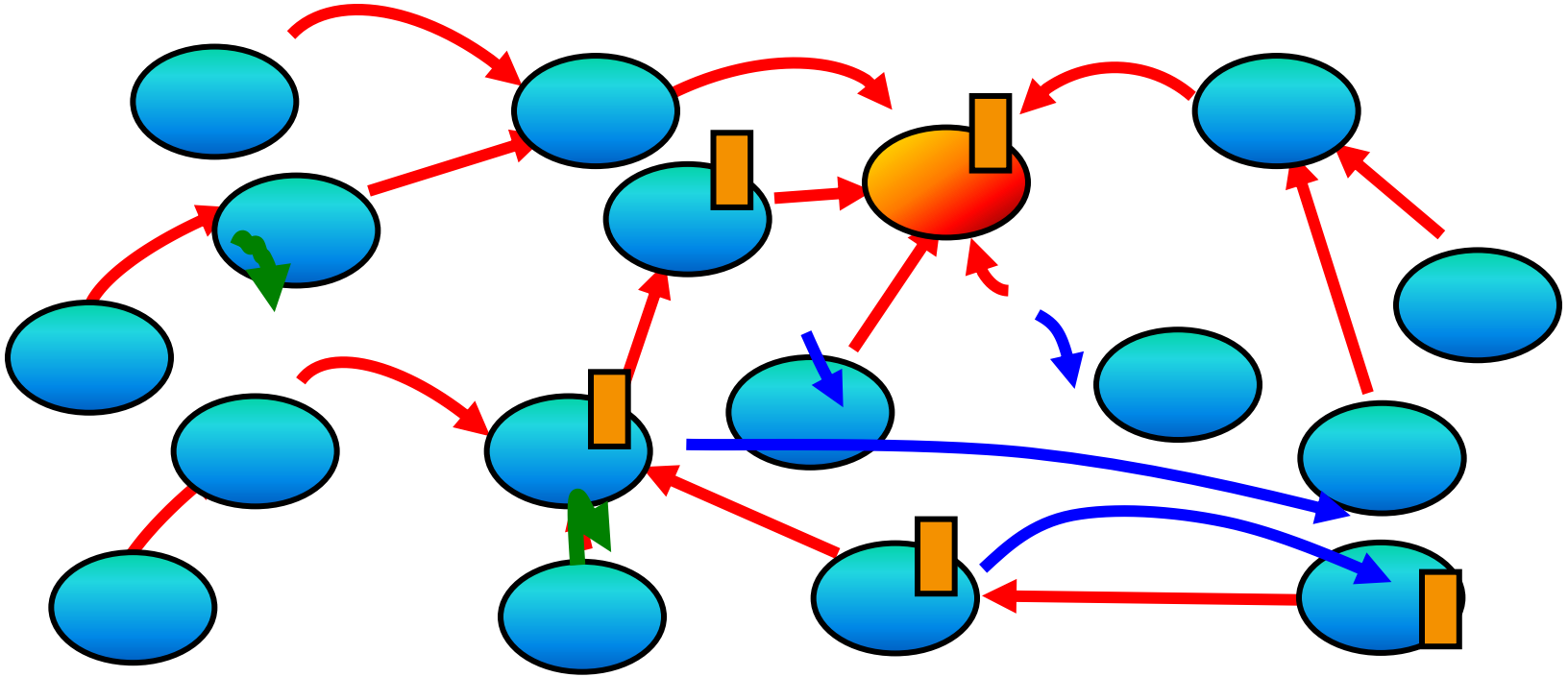
Surrogate Routing példa



Lokalizáció

- Egy szerver S bejelenti hogy rendelkezik az A állománnyal
  - S elküld egy Publish (ObjectID(A), ServerID(S)) üzenetet a Root(A) felé
  - Minden közbeeső router tárolja a linket (A→S)
- Query(A) → Root (A) felé
  - Ha útközben valaki tárolta a linket, a kérést azonnal továbbküldi a megfelelő helyre

Lokalizáció



Tapestry alkalmazások

- Mnemosyne
  - Biztonságos elosztott fájlrendszer
    - A Freenet-hez hasonló szolgáltatás
  - http://www.cs.rice.edu/Conferences/IPTPS02/107.pdf
- Bayeux
  - Alkalmazás rétegbeli multicast megoldás
    - Későbbi előadásban részletesen
- Spamwatch
  - Elosztott spam szűrő
  - A felhasználók címkézik a spam leveleket
  - Az e-mail szerverek egy Tapestry hálóba kötve
  - http://www.zhoufeng.net/eng/spamwatch/
- OceanStore
  - Elosztott tárolás
  - http://oceanstore.cs.berkeley.edu/

Összefoglalás

- Strukturálatlan vs strukturált P2P
- Tervezhető, algoritmizálható rendszer
- Nincs központi entitás - hurrá
  - És ez nem tetszik a felhasználóknak - búúú
- CAN – szép elméleti példa
- Kademlia és Tapestry – használ(t/j)ák
- Chord to come…