HY463 - Συστήματα Ανάκτησης Πληροφοριών
Information Retrieval (IR) Systems

Μέρος Γ

Συστήματα Ομοτίμων (Peer-to-Peer Systems) και Ανάκτηση Πληροφοριών

Γιάννης Τζίτζικας

Διάλεξη : 16-17
Ημερομηνία : 12,17-5-2006

Συστήματα Ομοτίμων
(Peer-to-Peer Systems)

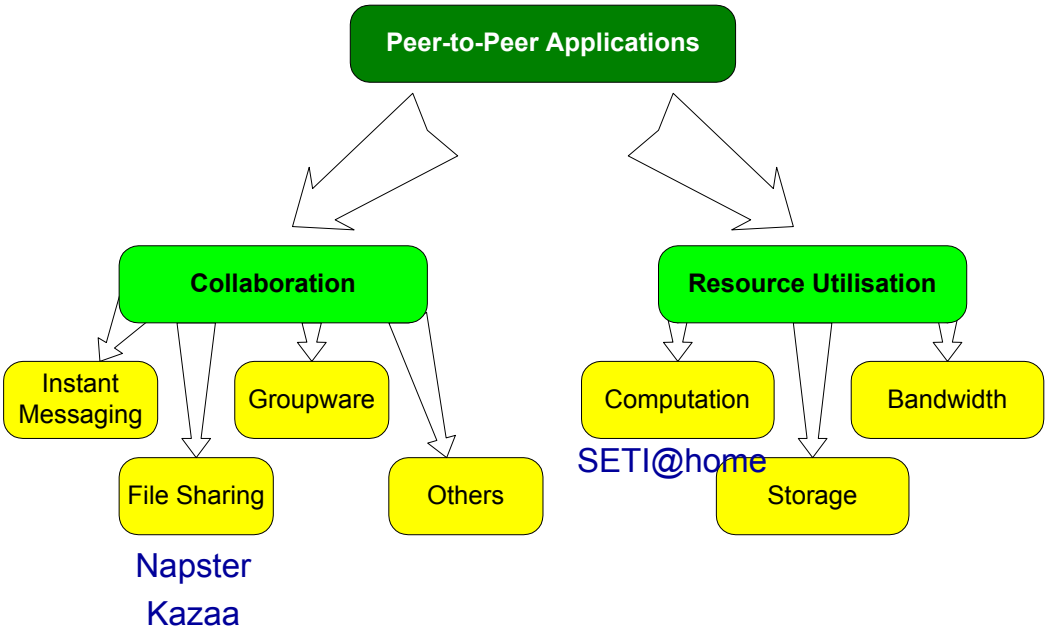
Διάρθρωση

- Κίνητρο
- Τύποι Ομότιμων Συστημάτων
  - Υβριδικά
  - Αποκεντρωμένα
  - Ιεραρχικά
  - Δομημένα
- Διαφορές με Κατανεμημένη Ανάκτηση
- Ομότιμα Συστήματα και Ανάκτηση Πληροφοριών

Ομότιμα Συστήματα: **Κίνητρο**

- Αξιοποίηση των ελεύθερων πόρων συστημάτων προσβάσιμων μέσω Internet για την επίλυση μεγάλων προβλημάτων (π.χ. SETI@home)
- δημιουργία συστημάτων πιο κλιμακόσιμων
- δημιουργία συστημάτων με μεγαλύτερη διαθεσιμότητα
- κατάργηση μονοπωλίων στην διάθεση της πληροφορίας
- αυτό-οργάνωση αντί κεντρικής διαχείρισης (και εξόδων αυτής)

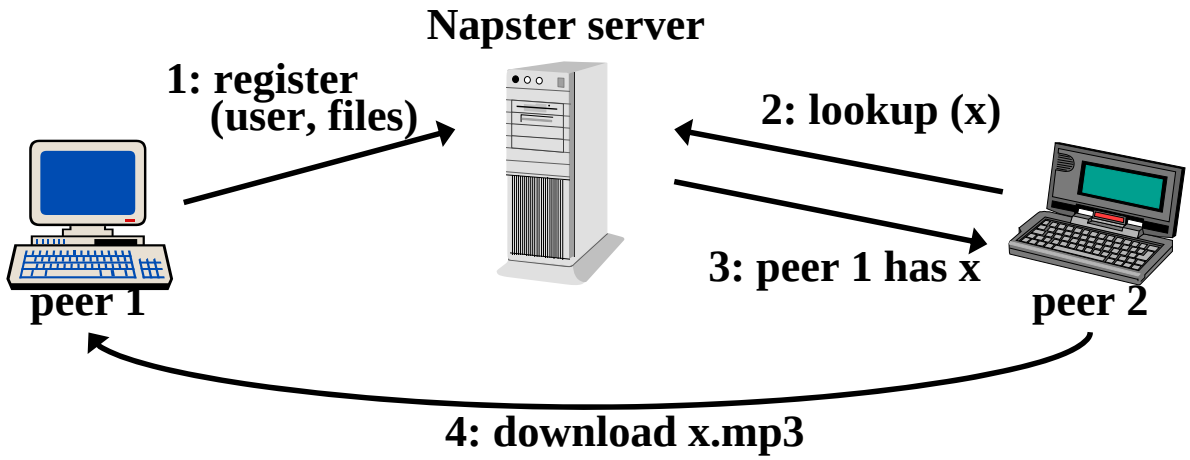
P2P: Εφαρμογές



Ομότιμα Συστήματα 1ης Γενιάς: Υβριδικά
Napster

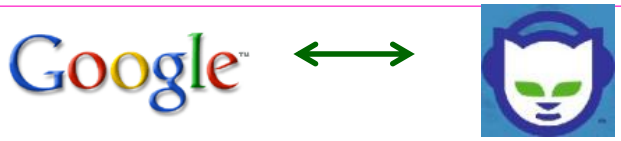
Ονομάζονται και **Υβριδικά Ομότιμα Συστήματα** (Hybrid P2P systems) διότι υπάρχει ένας κεντρικός εξυπηρετητής

Napster (1998-2001): διαμοιρασμός MP3



Μπορούμε να τα δούμε ως publish-subscribe systems: ο ιδιοκτήτης ενός αρχείο το διαθέτει με ένα όνομα x, οι άλλοι χρήστες μπορούν να αναζητήσουν το x, να βρουν ένα αντίγραφο και να το κατεβάσουν

Google (Client-Server) vs. Napster (P2P)



- Και οι δυο εφαρμογές έχουν την ίδια κλίματα
  - εκατομμύρια αναζητήσεις ημερησίως
  - Terabytes δεδομένων
- Google
  - Στηρίζεται σε περίπου 100.000 μηχανές
  - το στήσιμο μιας τέτοια εφαρμογής έχει μεγάλο κόστος (μόνο μια μεγάλη επιχείρηση μπορεί να κάνει τέτοια επένδυση)
- Napster
  - ο server χρησιμοποιεί μόνο 100 μηχανές
  - το κόστος αποθήκευσης και μεταφοράς των μουσικών αρχείων χρεώνεται στις μηχανές των χρηστών του συστήματος (γι' αυτό ονομάζεται P2P)
  - μικρό κόστος

Τα Πλεονεκτήματα των Ομοτίμων Συστημάτων

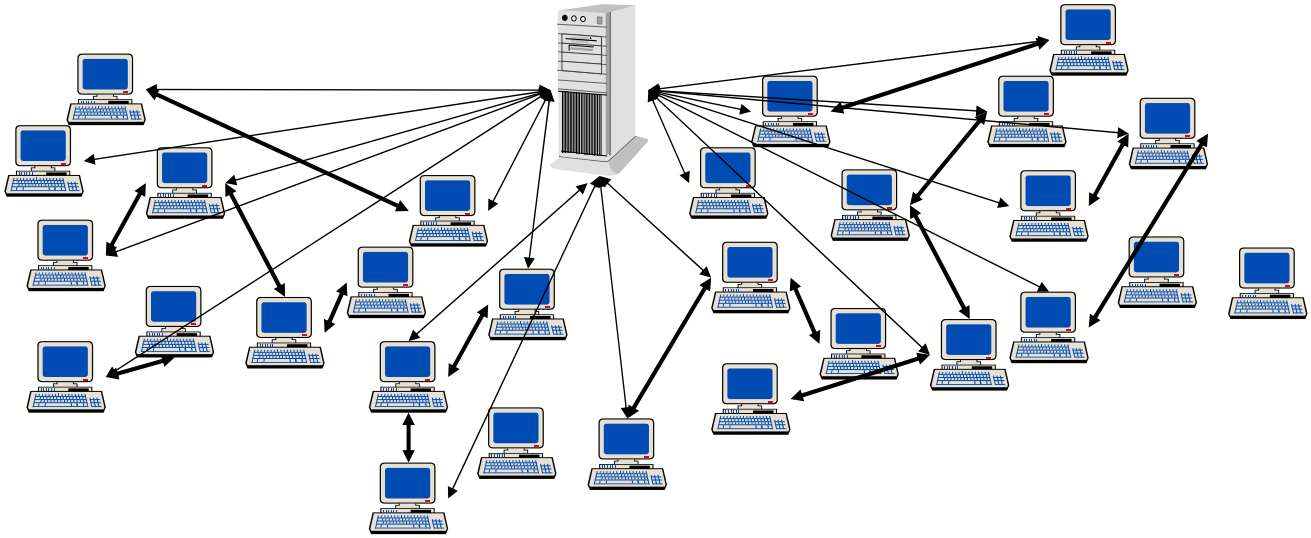
Διαμερισμός πόρων

- **αποθηκευτικών**
  - οι εκατομμύρια χρήστες του Napster αποθηκεύουν τα αρχεία, όχι ο εξυπηρετητής
- **επικοινωνίας**
  - το κατέβασμα αρχείων γίνεται μεταξύ των χρηστών, ο εξυπηρετητής δεν παρεμβάλλεται
- **εισαγωγής στοιχείων**
  - οι χρήστες του Napster εισάγουν τα αρχεία στο σύστημα
  - οι χρήστες του Napster τα κατηγοριοποιούν

Δίδαγμα:

Η αποκέντρωση επιτρέπει τη δημιουργία εφαρμογών παγκόσμιας κλίμακας χωρίς την ανάγκη μεγάλων επενδύσεων αλλά με την αξιοποίηση των πόρων που ήδη υπάρχουν

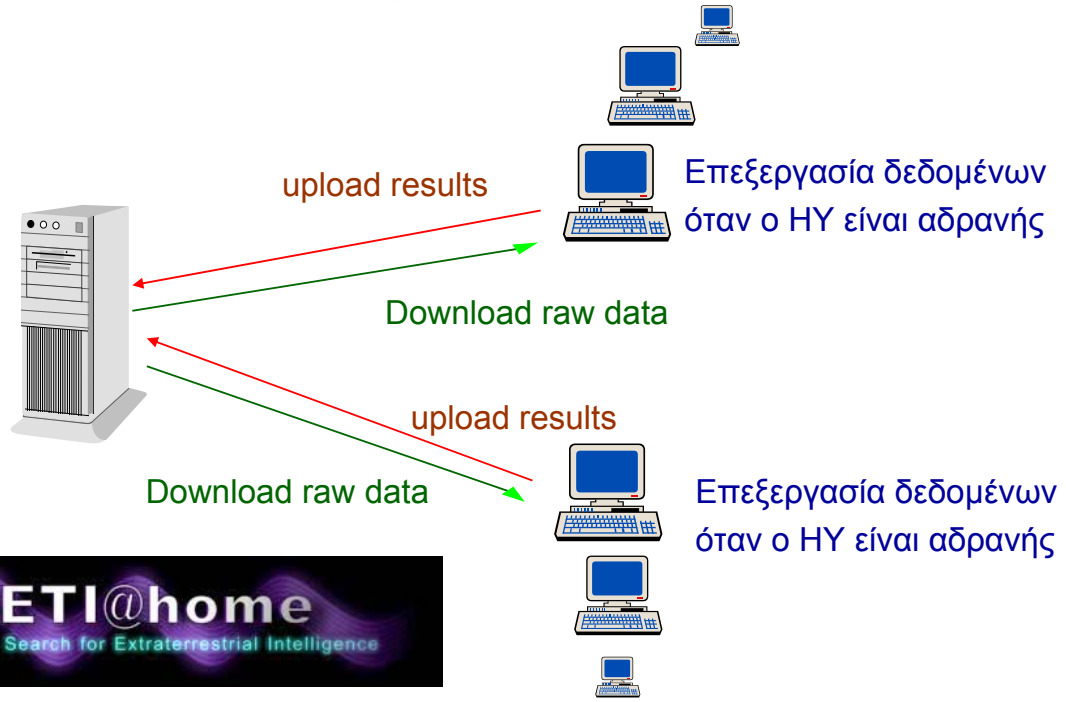
Ομότιμα Συστήματα 1ης Γενιάς: Υβριδικά
Napster



A central point of failure

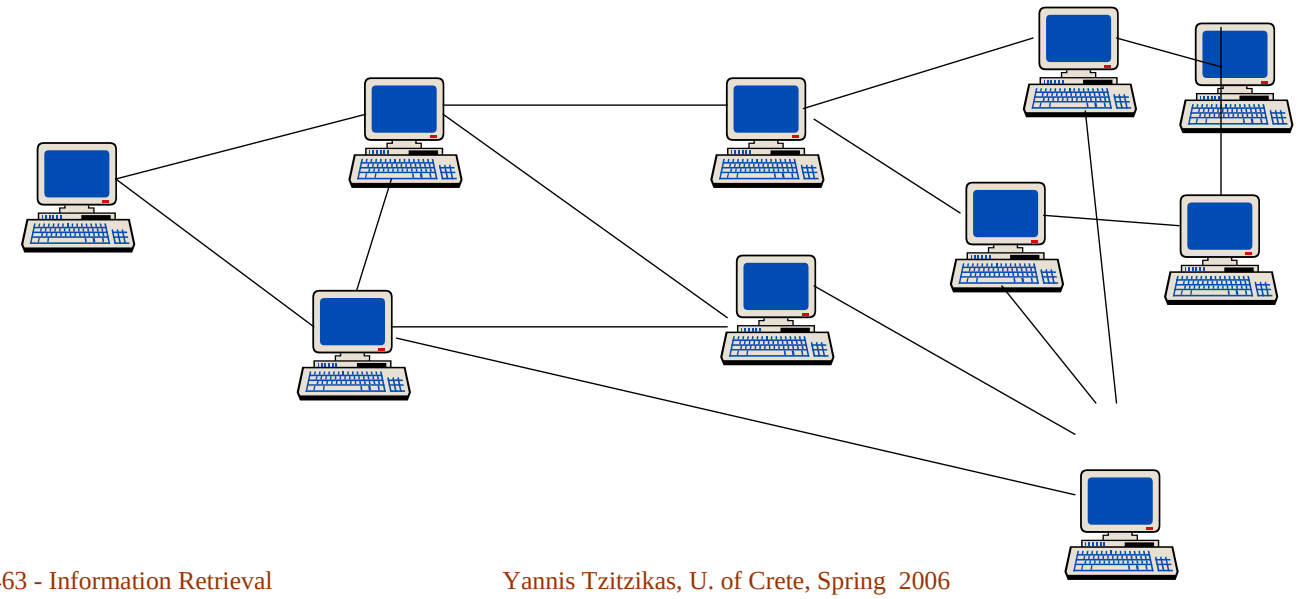
Ομότιμα Συστήματα 1ης Γενιάς: Υβριδικά
SETI@home

Σκοπός: **Διαμοιρασμός υπολογιστικών πόρων**
(αξιοποίηση των περιόδων αδράνειας των εγγεγραμμένων ΗΥ)



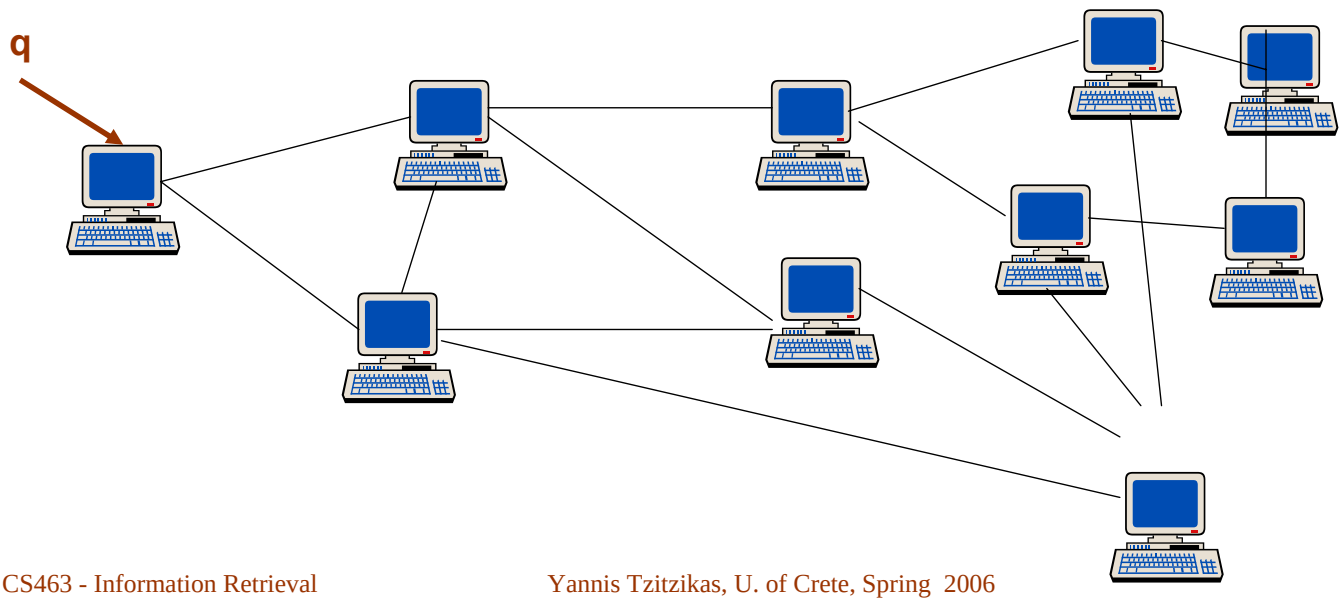
Ομότιμα Συστήματα 1ης Γενιάς: **Αποκεντρωμένα GNUTELLA**

- Δεν υπάρχει **κανένας κεντρικός εξυπηρετητής**
- Ονομάζονται και Αποκεντρωμένα (Decentralized P2P systems), Αδόμητα (Unstructured P2P systems), Pure P2P systems
- Gnutella (1999-now):

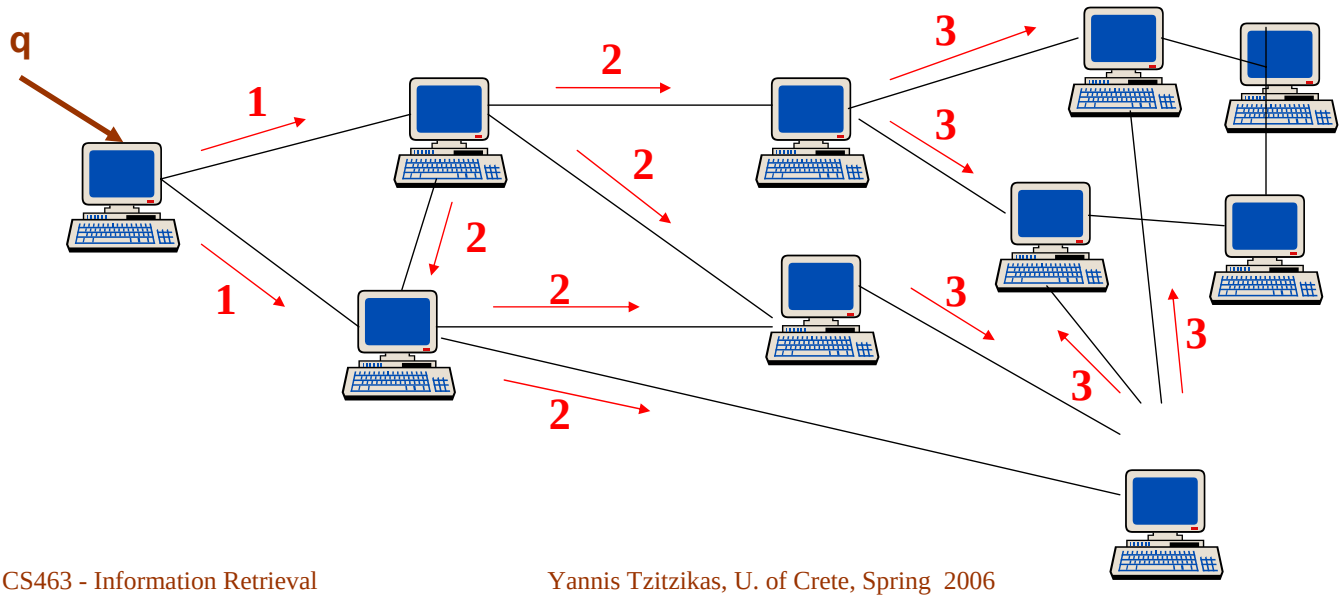


Ομότιμα Συστήματα 1ης Γενιάς: **Αποκεντρωμένα GNUTELLA**

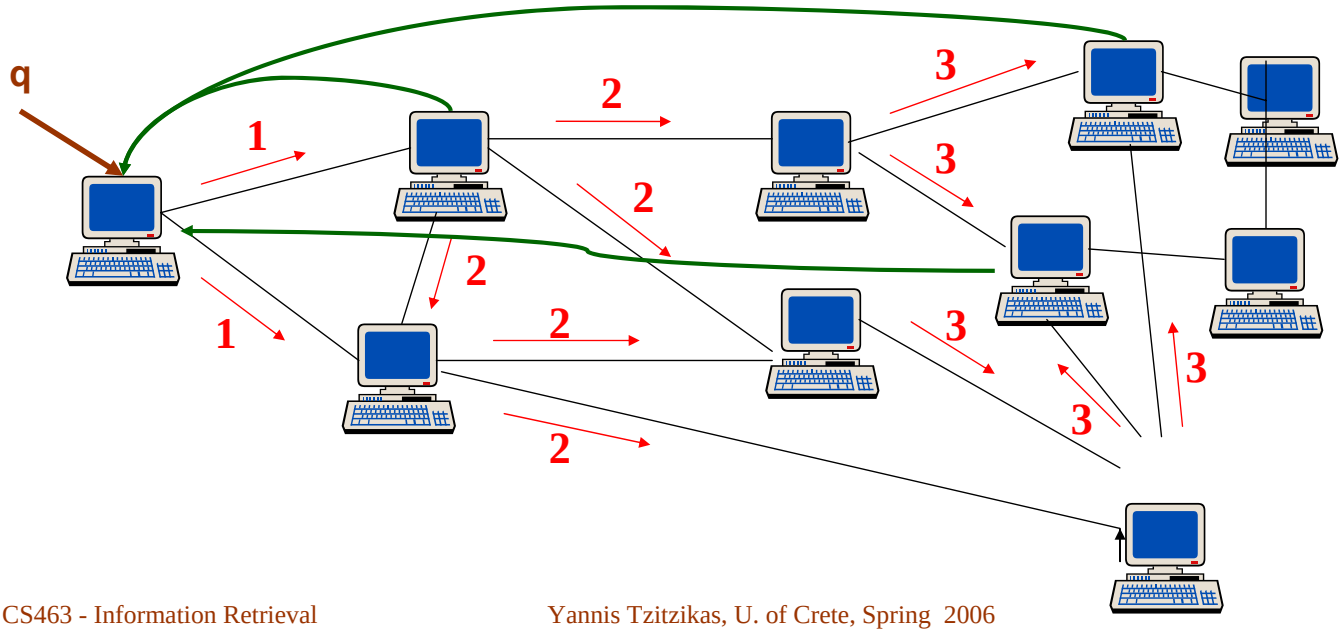
- Δεν υπάρχει **κανένας κεντρικός εξυπηρετητής**
- Ονομάζονται και Αποκεντρωμένα (Decentralized P2P systems), Αδόμητα (Unstructured P2P systems), Pure P2P systems
- Gnutella (1999-now):



Κατακλυσμός Μηνυμάτων (Message Flooding or Gossiping)

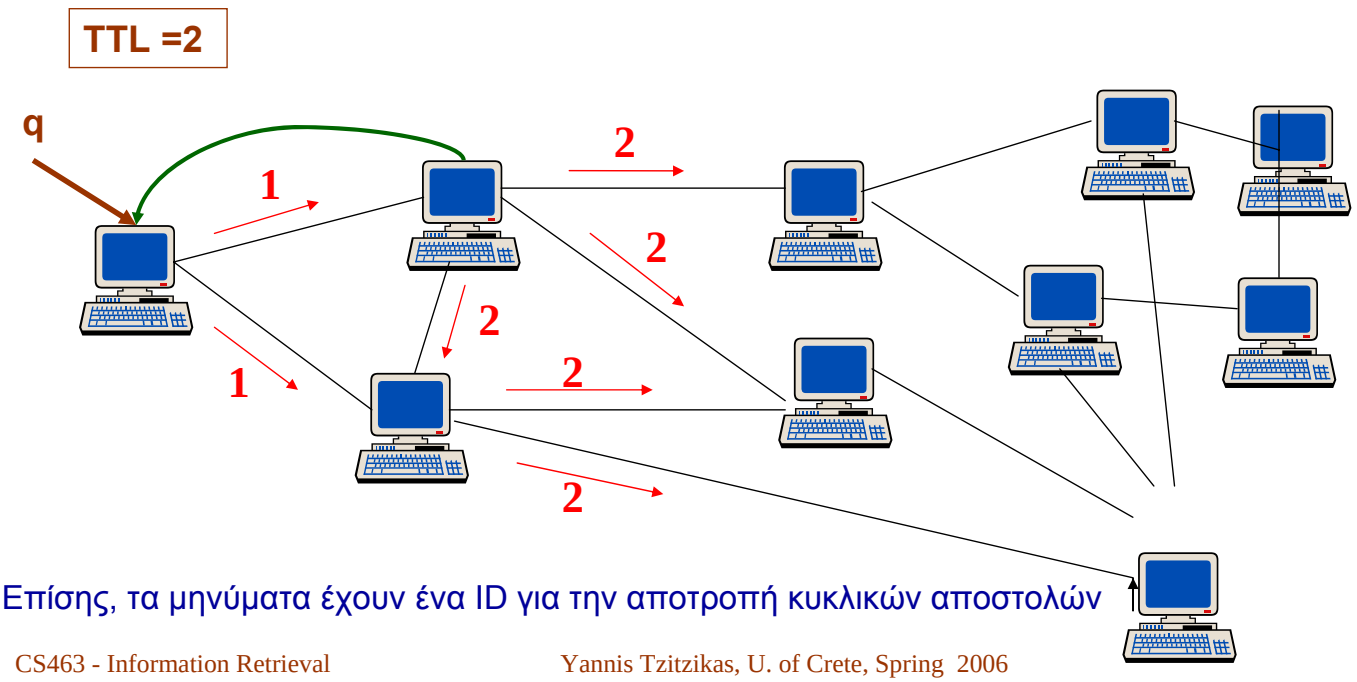


Κατακλυσμός Μηνυμάτων (*Message Flooding*)



Ομότιμα Συστήματα 1ης Γενιάς: **Αποκεντρωμένα GNUTELLA**

- Τα μυνήματα έχουν ένα TTL (time-to-live) tag

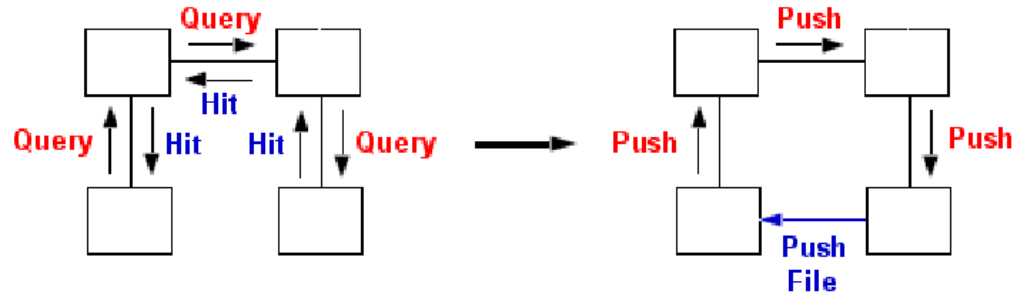


Επίσης, τα μηνύματα έχουν ένα ID για την αποτροπή κυκλικών αποστολών

GNUTELLA: The protocol

- **Ping**
  - Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.
- **Pong**
  - The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.
- **Query**
  - The primary mechanism for searching the distributed network. A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.
- **QueryHit**
  - The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.
- **Push**
  - A mechanism that allows a firewalled servent to contribute file-based data to the network.

GNUTELLA: The protocol



Example 2. Query/QueryHit/Push Routing

- Συνήθως κάθε κόμβος προωθεί μια επερώτηση σε C γείτονες (συνήθως C=3)
- Τυπική τιμή TTL=7
  - (πειράματα έδειξαν ότι η διάμετρος του Gnutella δικτύου είναι συνήθως 7)

Napster vs. Gnutella

- Napster (υπάρχει κεντρικός εξυπηρετητής)
  - single point of failure
  - στόχος νομικής επίθεσης
- Gnutella (δεν υπάρχει κεντρικός εξυπηρετητής)
  - δεν υπάρχει ”single point of failure”
  - δεν μπορεί να γίνει εύκολα στόχος νομικής επίθεσης
  - δεν απαιτεί καμία επένδυση
  - δεν έχει κόστος διαχείρισης (administration)
  - “self-organizing system”
    - however, “free-riders” may occur

GNUTELLA: Επιδόσεις

Επιδόσεις

- Χρόνος αναζήτησης: Σχετικά μικρός
- Πλήθος Μηνυμάτων: Μεγάλο
- Κόστος αποθήκευσης: Μικρό (κάθε κόμβος γνωρίζει μόνο τους διπλανούς του)
- Κόστος ενημέρωσης: Μικρό (γείτονες)
- Ανθεκτικότητα σε σφάλματα: Μεγάλη

Ομότιμα Συστήματα 2ης Γενιάς

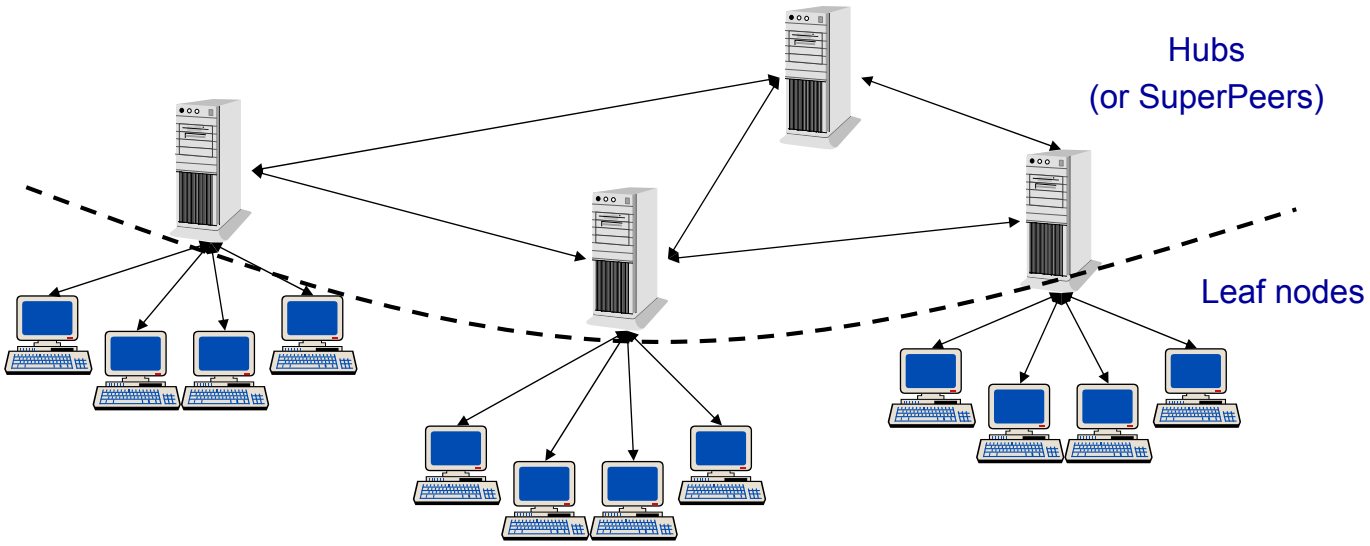
Πρωτότυπα ερευνητικά συστήματα:
Chord (MIT), CAN (Berkeley), OceanStore/Tapestry (Berkeley), Farsite (MSR), Spinglass/Pepper (Cornell), Pastry/PAST (Rice, MSR), Viceroy (Hebrew U), P-Grid (EPFL), P2P-Net (Magdeburg), Pier (Berkeley), Peers (Stanford), Kademlia (NYU), Bestpeer (Singapore), YouServ (IBM Almaden), Hyperion (Toronto), Piazza (UW Seattle), PlanetP (Rutgers), SkipNet (MSR),

Μπορούμε να διακρίνουμε 2 μεγάλες κατηγορίες

- Ιεραρχικά Ομότιμα Συστήματα (Hierarchical P2P systems)
  - Π.χ. Το σύστημα Kazaa
- Δομημένα Ομότιμα Συστήματα (Structured P2P systems)
  - Π.χ. το σύστημα Chord

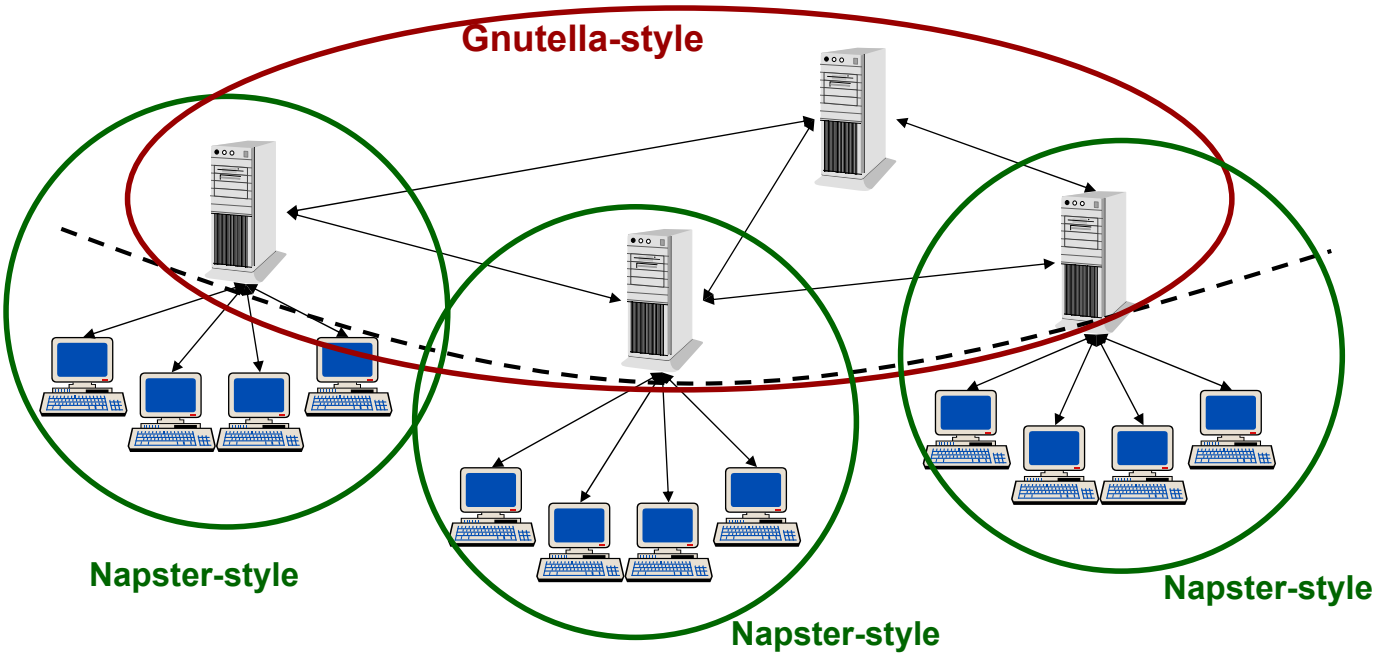
Ιεραρχικά Ομότιμα Συστήματα (**Hierarchical** P2P Systems)

Συστήματα: Morpheus, Kazaa, Limewire, JXTA Search, Gnutella 0.6



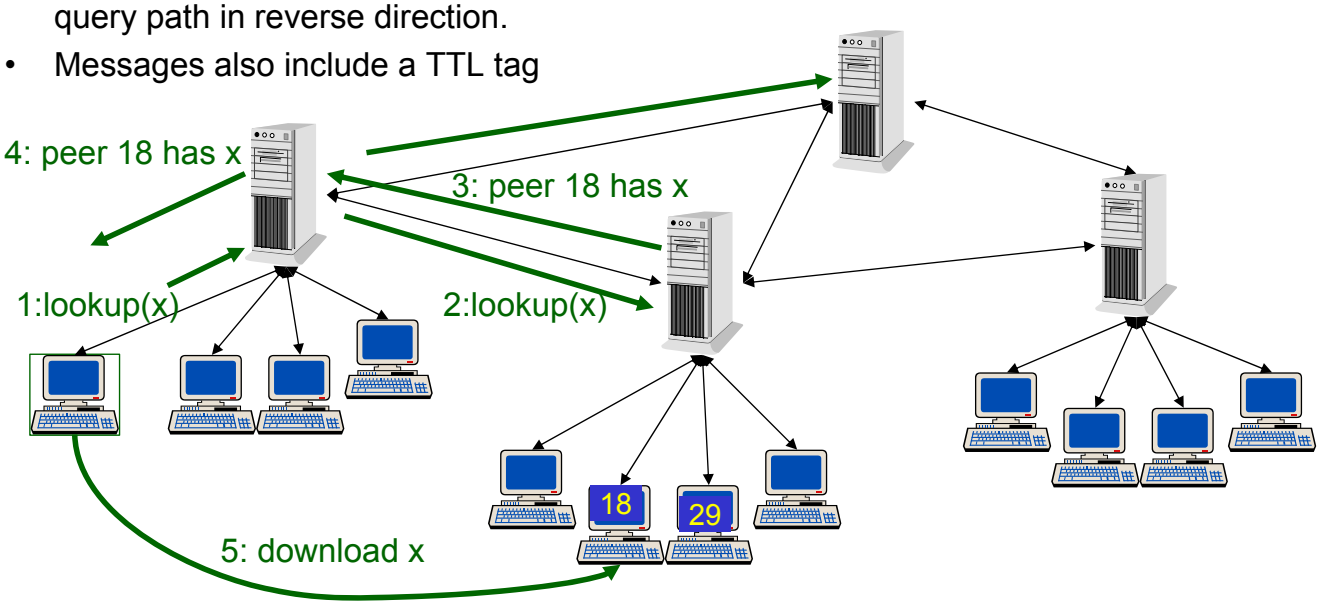
Ιεραρχικά Ομότιμα Συστήματα (**Hierarchical** P2P Systems)

Συνδυασμός Napster και Gnutella



Ιεραρχικά Ομότιμα Συστήματα (Hierarchical P2P Systems)

- Searching relies on **message-passing** between nodes.
- A query generated by a client node and is routed to a hub, from one hub to another, or from a hub to a leaf node.
- A response message ("queryhit") is generated by a leaf node and routed back along the query path in reverse direction.
- Messages also include a TTL tag



Ιεραρχικά Ομότιμα Συστήματα (Hierarchical P2P Systems)

Επιδόσεις

- Χρόνος αναζήτησης: Πολύ μικρός
- Πλήθος Μηνυμάτων: Μικρό
- Κόστος αποθήκευσης: Μικρό στα φύλλα, Μεγάλο στους εξυπηρετητές ευρετηρίου
- Κόστος ενημέρωσης: Μικρό
- Ανθεκτικότητα σε σφάλματα: Μικρή

Δομημένα Ομότιμα Συστήματα (**Structured** P2P Systems)

Σκοπός:

Γρήγορη εύρεση του κόμβου που περιέχει ένα κλειδί **χωρίς** τη χρήση **κεντρικού** εξυπηρετητή και ανταλλάσσοντας **λίγα** μηνύματα

- Εύκολο κομμάτι:
  - κατανομή ευρετηρίου σε όλους τους κόμβους
- Δύσκολο:
  - κατανομή ευρετηρίου σε όλους τους κόμβους με τέτοιο τρόπο ώστε να έχουμε **γρήγορη αναζήτηση**

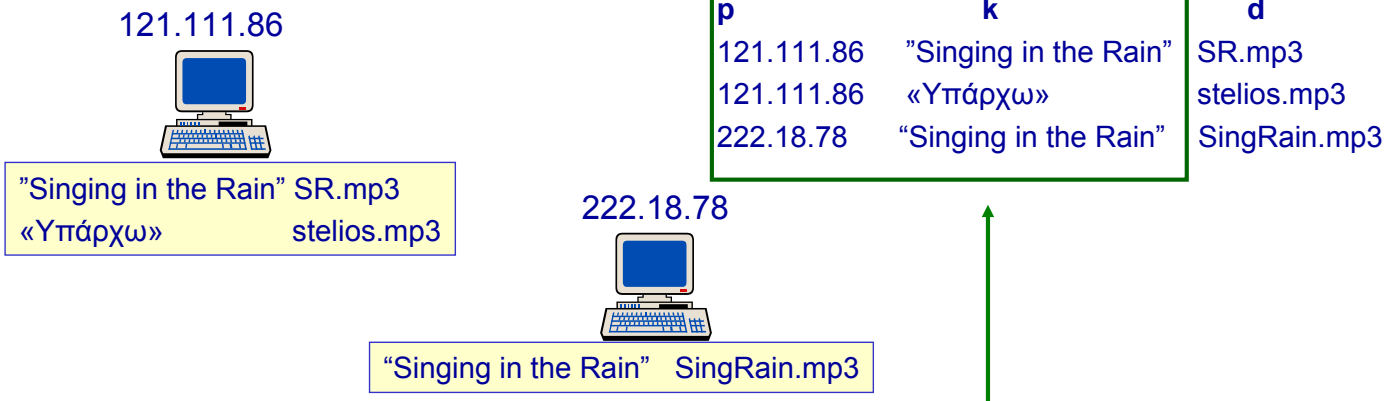
- Συστήματα
  - Freenet, Chord, CAN, Pastry, Tapestry, FreeNet, P-Grid,

Δομημένα Ομότιμα Συστήματα

- Κοινά χαρακτηριστικά των δομημένων ομότιμων συστημάτων
  - **κάθε** κόμβος διατηρεί ένα μικρό τμήμα του καθολικού ευρετηρίου (πίνακας δρομολόγησης)
  - οι αναζητήσεις γίνονται με προώθηση μηνυμάτων προς τη «σωστή» κατεύθυνση

- Διαφορετικές Προσεγίσεις
  - FreeNet: caching πληροφορίας ευρετηρίου κατά μήκος των μονοπατιών αναζήτησης
  - Chord: κατασκευή ενός κατανεμημένου πίνακα κατακερματισμού (Distributed Hash Table, DHT)
  - CAN: Δρομολόγηση βάσει d-διάστατου χώρου
  - P-Grid: κατανομή ενός δυαδικού δένδρου αναζήτησης

Το Πρόβλημα του Εντοπισμού Πόρου
(Resource Location)



Έστω peer με δνση **p** που αποθηκεύει στοιχείο **d** που χαρακτηρίζεται από το κλειδί **k**

Ζητούμενο: Δοθέντος **k** (ή συνθήκης πάνω στο **k**) εντόπισε τον peer που έχει το **d**, δηλαδή βρες το ζεύγος ευρετηρίου **(k,p).**

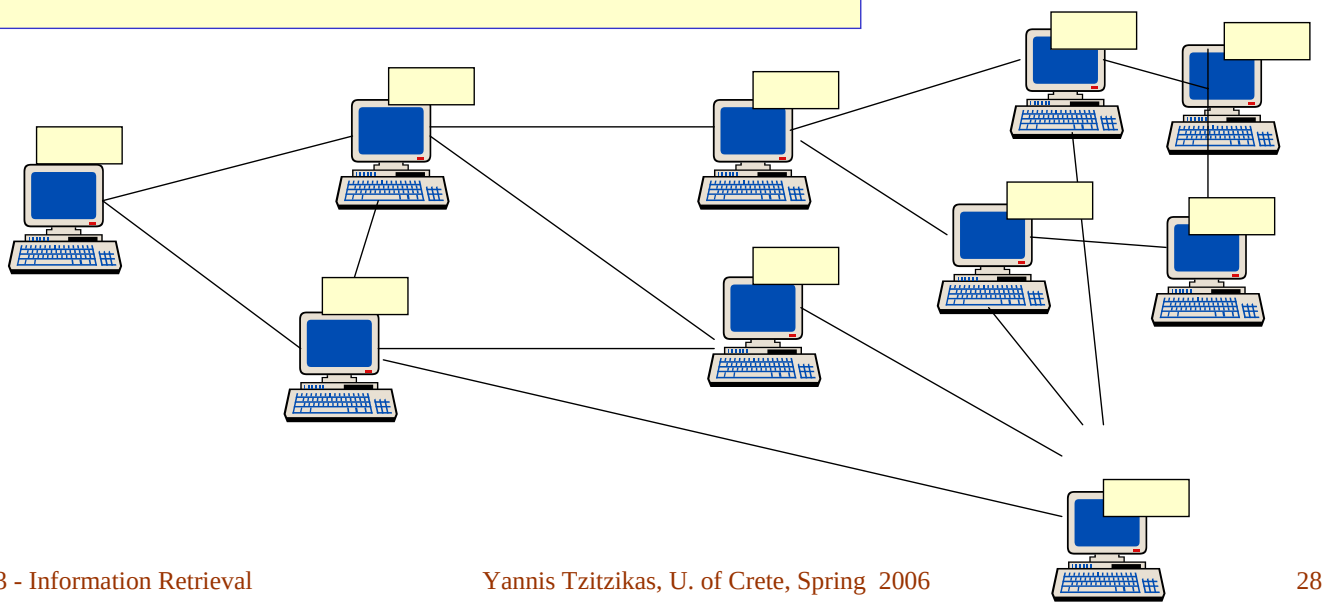
(άρα το ευρετήριο μας αποτελείται από ζεύγη της μορφής **(k,p))**

*Κρίσιμο ερώτημα: Πως μπορούμε να (α) **φτιάξουμε**, (β) **συντηρήσουμε** και (γ) να **χρησιμοποιήσουμε** ένα τέτοιο ευρετήριο χωρίς κεντρικό έλεγχο;*

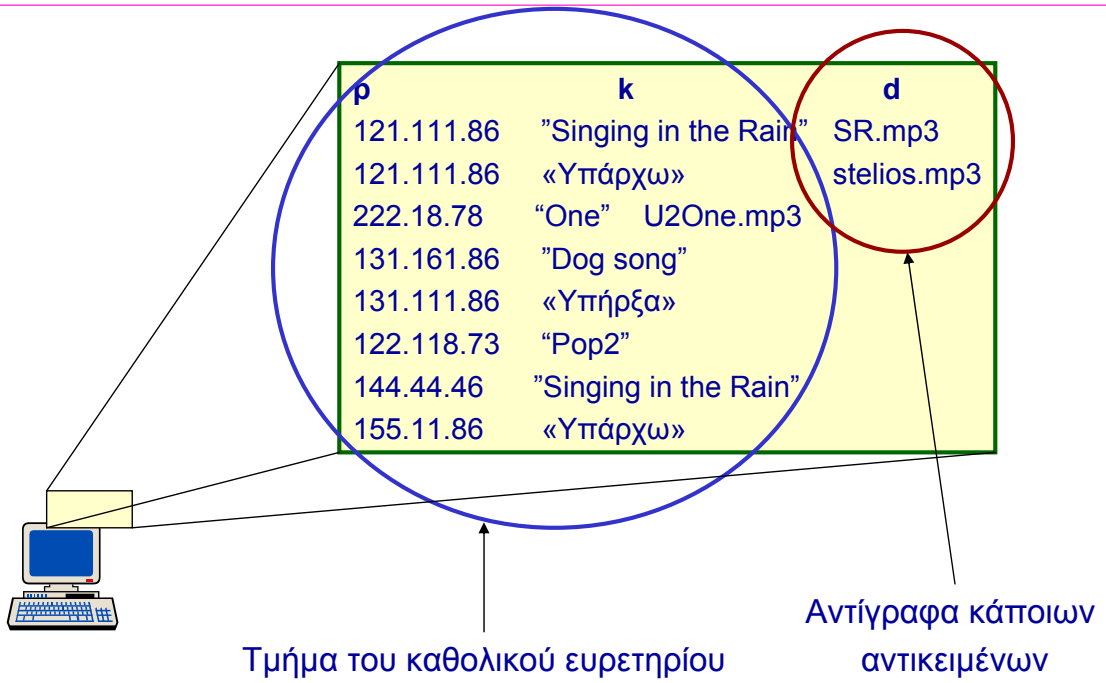
Freenet:

- Σύστημα για δημοσίευση και ανάκτηση δεδομένων με έμφαση στην ανωνυμία (και των συγγραφέων και των αναγνωστών)
- Τα κλειδιά και τα δεδομένα αποθηκεύονται *κρυπτογραφημένα*

Μοιάζει με: Gnutella + cache at each node



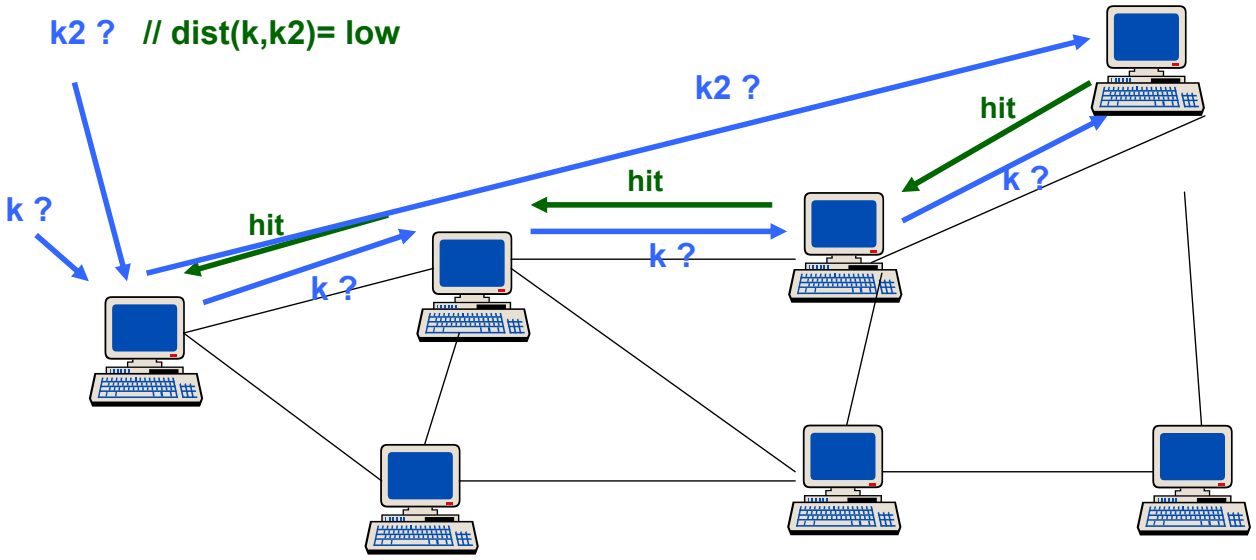
Freenet: Cache



Freenet: Τρόπος Εντοπισμού Πόρου

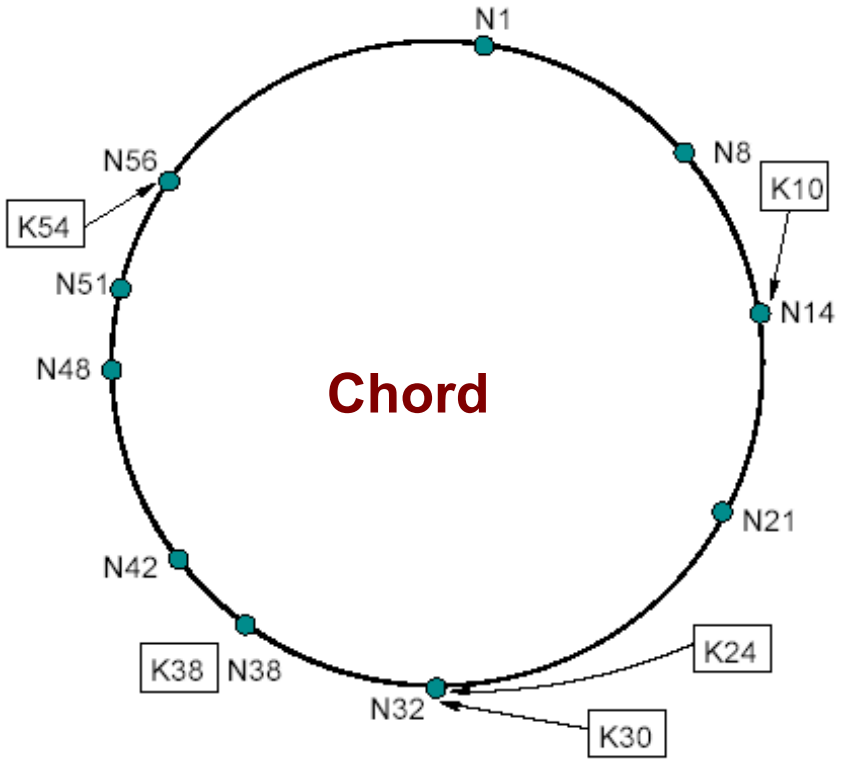
- Έλευση επερώτησης k
  - Αν η έγγραφή (p,k,d) είναι στη κρυφή μνήμη επέστρεψε το d
  - Αλλιώς προώθησε την επερώτηση στον κόμβο που έχει το **πιο όμοιο κλειδί**
- Η διαδικασία αυτή συνεχίζεται με αυτόν τον τρόπο έως ότου ευρεθεί το αναζητούμενο ή το TTL φτάσει την τιμή 0.
- Έλευση απάντησης (k,p,d)
  - Η τριάδα εισάγεται στην κρυφή μνήμη
  - Η παλαιότερη εγγραφή (least recently used) διαγράφεται από την κρυφή μνήμη
- Παρατηρήσεις
  - Οι δρομολογήσεις που κάνουν οι κόμβου βελτιώνονται συν το χρόνο
  - Οι κόμβοι τείνουν να έχουν στην κρυφή τους μνήμη εγγραφές με παρόμοια κλειδιά (άρα επιτυγχάνεται ένα είδος ομαδοποίησης)

Freenet



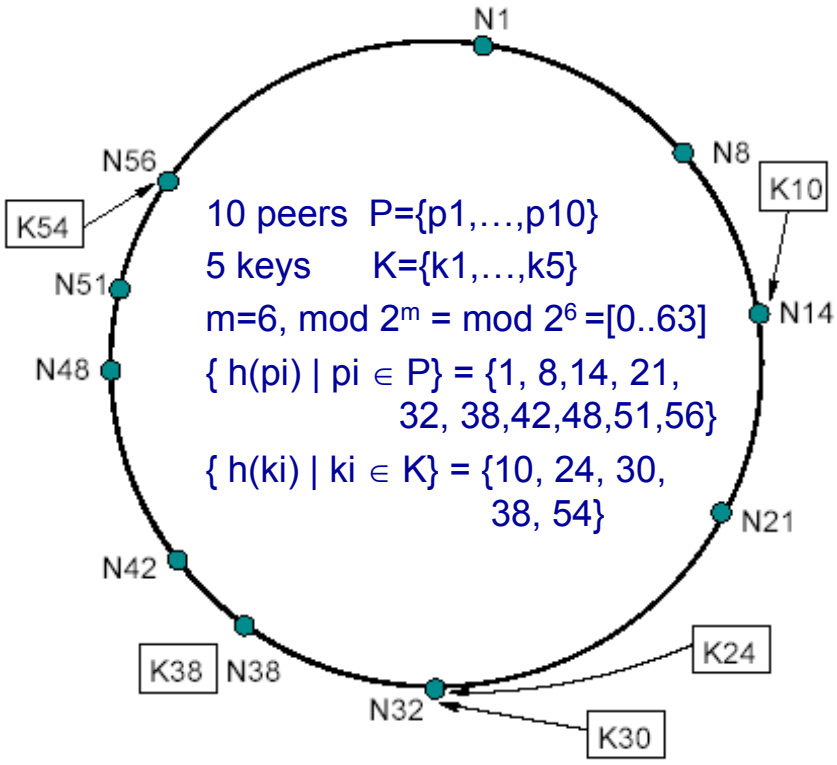
Freenet: Εισαγωγή Νέου Πόρου

- Υπολογίζουμε το κλειδί του νέου πόρου
- Φτιάχνεται η εγγραφή εισαγωγής και στέλνεται στον γείτονα με το πιο κοντινό κλειδί
- Κάθε κόμβος που λαμβάνει το νέο κλειδί, ελέγχει αν το κλειδί αυτό υπάρχει ήδη
- αν ναι, έχουμε σύγκρουση (collision), και άρα ο αρχικός κόμβος πρέπει να προτείνει ένα νέο κλειδί
- αν όχι, δρομολόγηση στον επόμενο κόμβο με τον ίδιο τρόπο
- Αν TTL=0 και δεν είχαμε καμία σύγκρουση, τότε η τριάδα αποθηκεύεται σε όλους τους κόμβους του μονοπατιού που ακολουθήθηκε



Chord (Distributed Hash Tables (DHT))

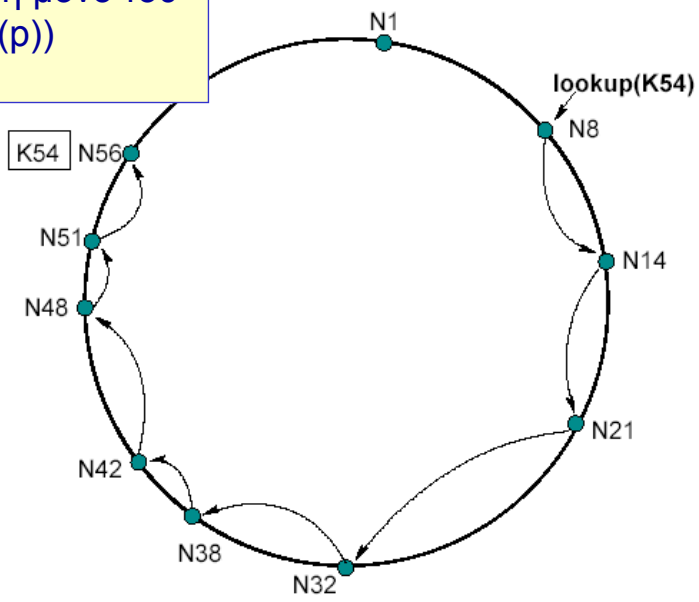
- Κατακερματισμός (Hashing) κλειδιών (k)και διευθύνσεων (p) σε δυαδικά κλειδιά με **m**-bits
  - π.χ. m=6, **h**(«υπάρχω»)=**11**, **h**(196.178.0.1)=**3**
- Τα δυαδ. κλειδιά τοποθετούνται σε έναν κύκλο modulo 2^m
  - Για m=8, κυκλική διάταξη των αριθμών 0…255
- Ένα κλειδί **k** εκχωρείται στον πρώτο κόμβο **p** τ.ω. h(**p**) >= h(**k**)
- Αυτός ο κόμβος λέγεται **successor(k)**



Chord : Απλός τρόπος εντοπισμού κόμβων

Έστω ότι κάθε κόμβος p ξέρει την δνση μόνο του επόμενου του (του p' με h(p') > h(p))

```
// ask node n to find the successor of id
n.find_successor(id)
  if (id  in   (n; successor])
     return successor;
  else
     // forward the query around the   circle
   return successor.find_successor(id);
```



=> Number of messages linear in the number of nodes !

Chord : Ένας πιο γρήγορος τρόπος εντοπισμού κόμβων με Πίνακες Δρομολόγησης

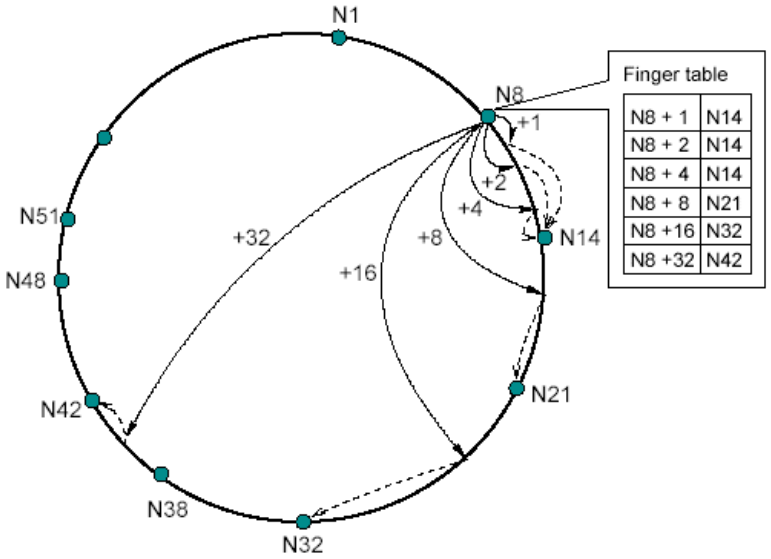
- Επιπλέον πληροφορία δρομολόγησης για επιτάχυνση
- Κάθε κόμβος n έχει έναν **πίνακα δρομολόγησης** με m εγγραφές
  - οι m αυτοί κόμβοι έχουν εκθετικά αυξανόμενη απόσταση από τον n
- Η i εγγραφή του πίνακα έχει την δνση του πρώτου κόμβου με κλειδί μεγαλύτερο ή ίσο με $n+2^{i-1}$

$$finger[i] = successor\ (n + 2^{i-1})$$

Chord : Παράδειγμα Πίνακα Δρομολόγησης

Finger table:

finger[i] = successor (n + 2^{i-1})



n=8

finger[1] = succ(n+1) = succ(9) = 14

finger[2] = succ(n+2) = succ(10) = 14

finger[3] = succ(n+4) = succ(12) = 14

finger[4] = succ(n+8) = succ(16) = 21

finger[5] = succ(n+16) = succ(24) = 32

finger[6] = succ(n+32) = succ(40) = 42

Chord: Εντοπισμός Πόρου με Πίνακες Δρομολόγησης

Έστω μια επερώτηση k προς έναν κόμβο n

Ο n κοιτάζει τον πίνακα δρομολόγησης του και

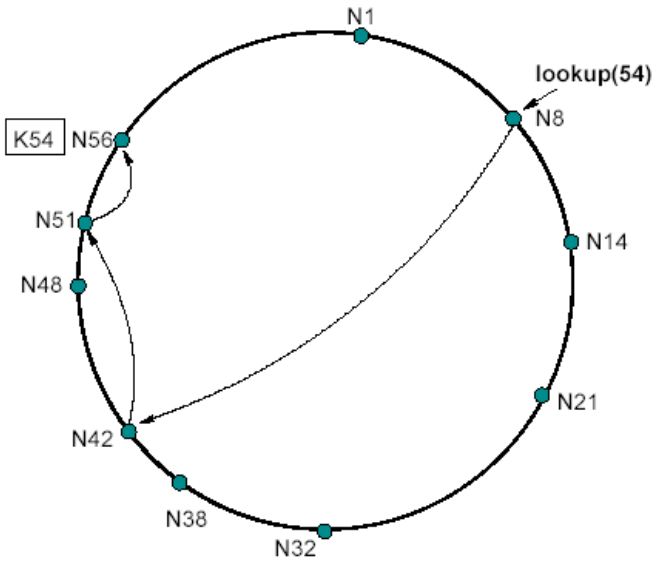
βρίσκει τον μικρότερο peer με κλειδί μεγαλύτερο αυτού της επερώτησης.

Αν δεν υπάρχει τέτοιος peer, τότε ο ίδιος είναι υπεύθυνος για το κ (και άρα το ζητούμενο βρέθηκε)

Αλλιώς προωθεί την επερώτηση

Αφού οι εγγραφές των πινάκων δρομολόγησης είναι εκθετικά αύξουσες, η αναζήτηση (με μεγάλη πιθανότητα) λαμβάνει λογαριθμικό χρόνο.

Chord: Πλήθος μηνυμάτων: O(log N)



N8

Finger table	
N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42

N42

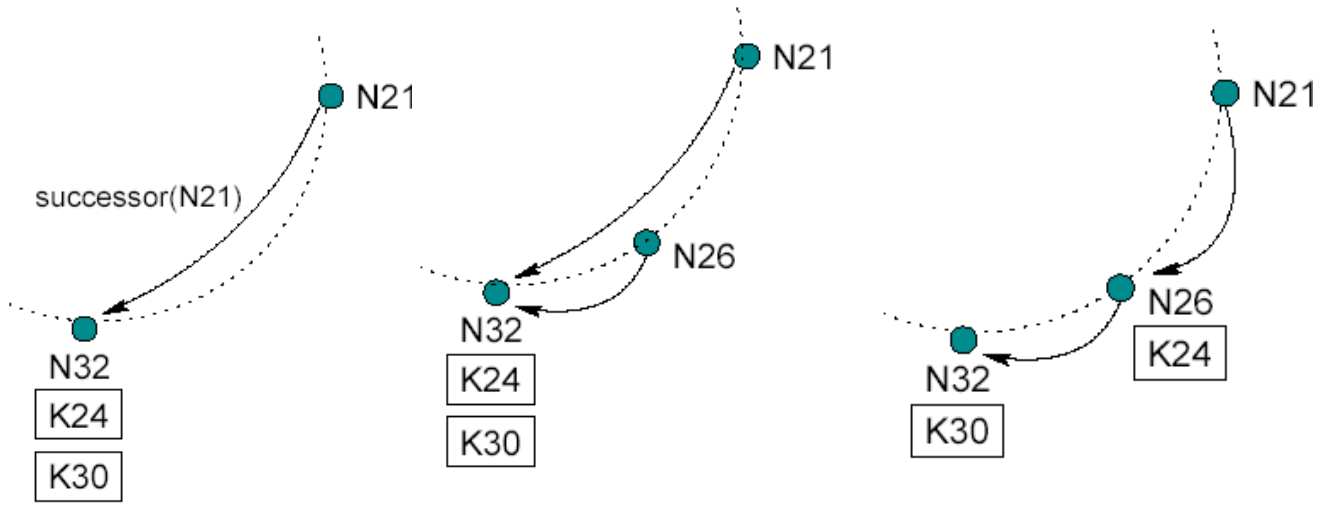
Finger table	
N42+ 1	N48
N42+ 2	N48
N42+ 4	N48
N42+ 8	N51
N42+16	N1
N42+32	N14

Εύρεση με ανταλλαγή τριών μηνυμάτων

- Search in finger table for the nodes which most immediatly precedes id
- Invoke find_successor from that node
- **=> Number of messages O(log N)**

Chord: Είσοδος νέου κόμβου

- Ο νέος κόμβος πρέπει να φτιάξει τον πίνακα δρομολόγησης του
- Το κόστος κατασκευής του είναι αυτό της αναζήτησης
- Οι άλλοι κόμβοι πρέπει να ενημερώσουν τους δικούς τους πίνακες



Chord : Περίληψη

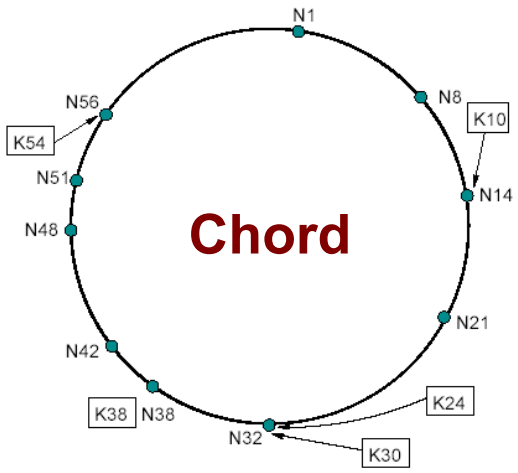
Βασικά σημεία:

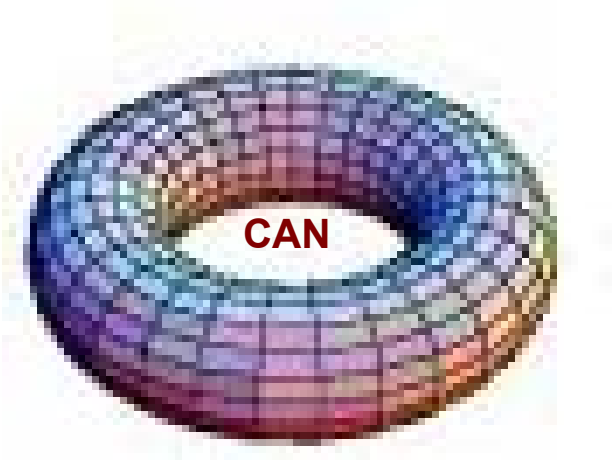
- Κάθε κόμβος αποθηκεύει πληροφορία για μικρό αριθμό κόμβων (m)
  - Κάθε κόμβος ξέρει περισσότερα για τους κοντινούς του όρους (απ'ότι για τους μακρινούς)

Επιδόσεις

- Χρόνος αναζήτησης: O(log n) (με μεγάλη πιθανότητα)
- Πλήθος Μηνυμάτων: O(log n) (επιλεκτική δρομολόγηση μηνυμάτων)
- Κόστος αποθήκευσης: O(log n) (πίνακας δρομολόγησης)
- Κόστος εισόδου/εξόδου κόμβου: O(log^2 n)
- Κόστος ενημέρωσης: μικρό (περίπου σαν το κόστος αναζήτησης)

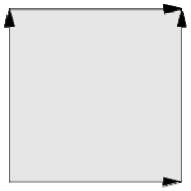
- Chord software
  - 3000 lines of C++ code, Library to be linked with the application, provides a lookup(key) – function: yields the IP address of the node responsible for the key, Notifies the node of changes in the set of keys the node is responsible for

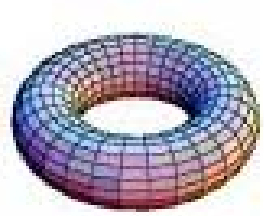




Δομημένα Ομότιμα Συστήματα
CAN (Content Addressable Network)

- Βασίζεται στον κατακερματισμό κλειδιών στον **κ-διάστατο** Καρτεσιανό χώρο (torus) (συνήθως κ=2-10)
  - Κλειδί = σημείο του κ-διάστατου χώρου
    - κ διαστάσεις, Hash(key) = (x1, …, xk)
  - Κάθε κόμβος είναι υπεύθυνος για ένα κομμάτι του χώρου, μία **ζώνη**
    - Αποθηκεύει το ευρετήριο των αντικειμένων των οποίων οι συντεταγμένες εμπίπτουν στην ζώνη του
  - Κάθε κόμβος αποθηκεύει τις διευθύνσεις των κόμβων των διπλανών ζωνών
  - Εύρεση πόρου = δρομολόγηση στις ζώνες

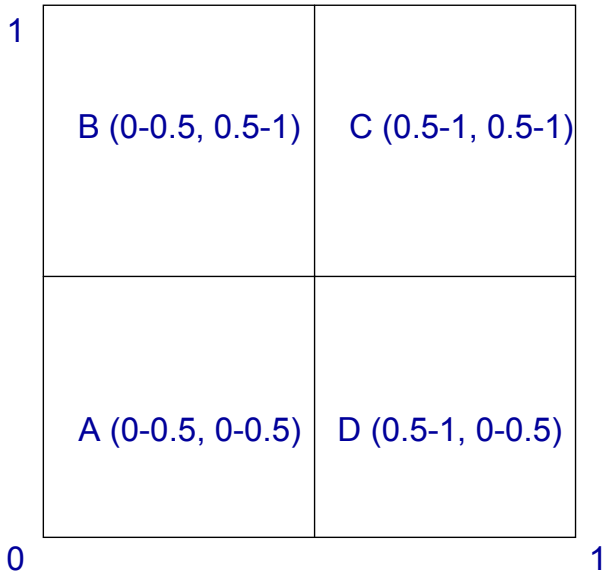




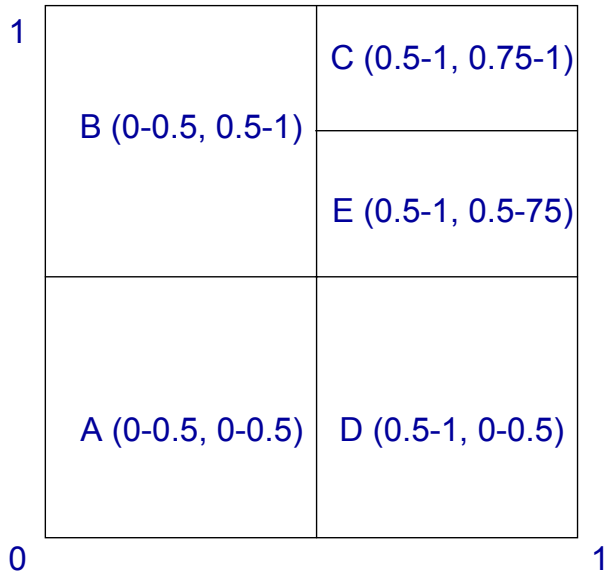
http://mathworld.wolfram.com/Torus.html

Δομημένα Ομότιμα Συστήματα
CAN

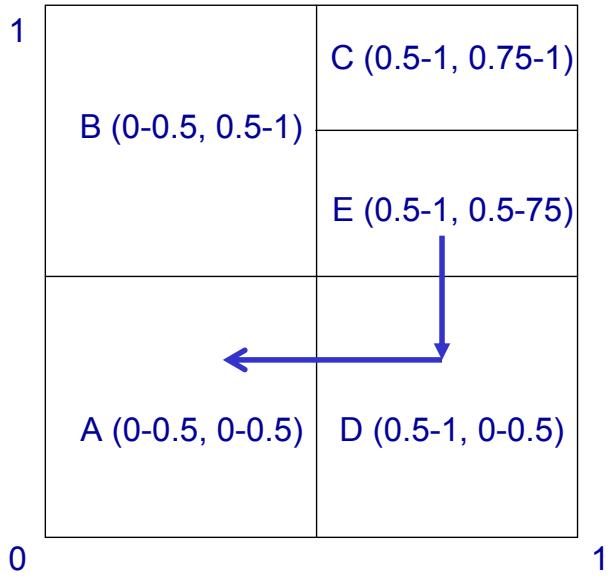
- Π.χ. για 2D, 4 peers A, B, C, D

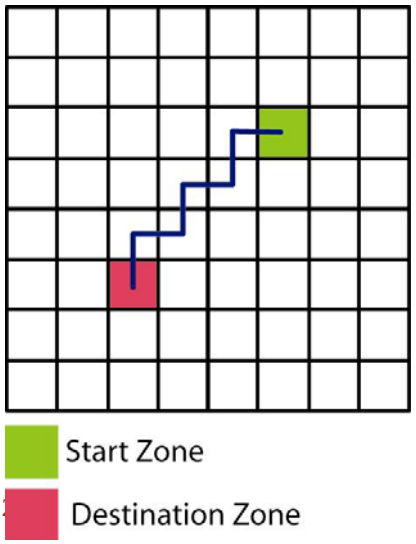


- Είσοδος ενός νέου κόμβου E

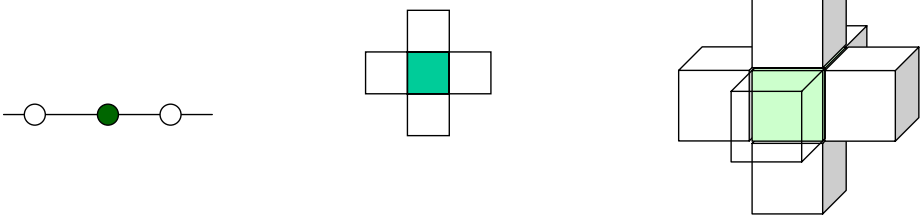


- Ο E θέλει να ανακτήσει το αντικείμενο με συντεταγμένες (0.2, 0.2)





- Αυξάνοντας τις διαστάσεις
  - **μειώνεται** το μήκος του μονοπατιού αναζήτησης
  - **αυξάνεται** το πλήθος των γειτόνων που πρέπει κάθε κόμβος να αποθηκεύει



- Πολυπλοκότητα αναζήτησης n κόμβοι, κ διαστάσεις

$$O(k\sqrt[k]{n})$$

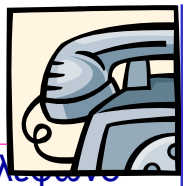
Βασικά σημεία:

- Κάθε κόμβος αποθηκεύει πληροφορία για ένα τμήμα του διανυσματικού χώρου και γνωρίζει τις δνσεις των διπλανών του κόμβων

Επιδόσεις

- Χρόνος αναζήτησης: O(k $n^{1/k}$) (με μεγάλη πιθανότητα)
- Πλήθος Μηνυμάτων: O(k $n^{1/k}$) (επιλεκτική δρομολόγηση μηνυμάτων)
- Κόστος αποθήκευσης: O(k) (πίνακας δρομολόγησης)
- Κόστος ενημέρωσης: μικρό (περίπου σαν το κόστος αναζήτησης)

Περίληψη Ομότιμων Συστημάτων

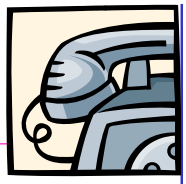


Έστω ότι στην Ελλάδα κατοικούν 1000 άτομα και κάθε ένας τους έχει ένα τηλ

Τρόποι εύρεσης του τηλεφώνου ενός κυρίου Χ

- Napster-style
  - Εύρεση τηλεφώνου τηλεφωνώντας στο 11811 του ΟΤΕ
- Gnutella-style
  - Εύρεση τηλεφώνου ρωτώντας όποιον βρούμε μπροστά μας (κ.ο.κ)
- Kazaa-style
  - Δεν υπάρχει ΟΤΕ για όλη την Ελλάδα, αλλά κάθε νομός έχει έναν τοπικό ΟΤΕ. Τηλεφωνούμε στον τοπικό και αν αυτός δεν το έχει, επικοινωνεί με τους υπόλοιπους τοπικούς ΟΤΕ

Περίληψη Ομότιμων Συστημάτων



- Freenet-style
  - Κάθε ένας έχει μια ατζέντα περιορισμένου μεγέθους. Εύρεση τηλεφώνου τηλεφωνώντας σε αυτόν που έχει το πλησιέστερο όνομα (π.χ. λεξικογραφικά), κ.ο.κ. Όταν εν τέλει βρεθεί, ενημερώνουμε την ατζέντα μας.
- Chord-style
  - Κάθε κάτοικος έχει μια ατζέντα με 10 τηλέφωνα (10 = log 1024)
  - Η εύρεση του τηλεφώνου του κυρίου Χ θα γίνει με 10 τηλεφωνήματα
- CAN-style
  - Κάθε ένας ξέρει το τηλέφωνο των γειτόνων του
    - αν όλοι οι Έλληνες ζουν σε μονοκατοικίες τότε κάθε ένας έχει 4 γείτονες (Βορ,Νοτ,Α,Δ)
    - αν όλοι οι Έλληνες ζουν σε 1 πολυκατοικία τότε κάθε ένας έχει 6 γείτονες
  - Για να τηλεφωνήσω σε κάποιον πρέπει να ξέρω που είναι το σπίτι του και τηλεφωνώ στο γείτονα μου που είναι προς εκείνη την κατεύθυνση (κ.ο.κ)
  - Αν όλοι μένουν σε μονοκατοικίες τότε 2 * SQRT(1000) = 64 τηλεφωνήματα
  - Αν όλοι μένουν σε μια πολυκατοικία τότε 3 * CubicRoot(1000) = 3 * 10 τηλεφωνήματα

Ανάκτηση Πληροφοριών & Συστήματα Ομοτίμων
(Peer-to-Peer Systems)


& IR

Lecture : 17

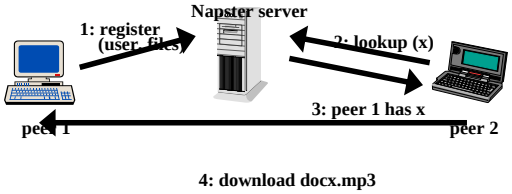
Τι διαφέρει η Ανάκτηση σε P2P συστήματα από την Κατανεμημένη Ανάκτηση;

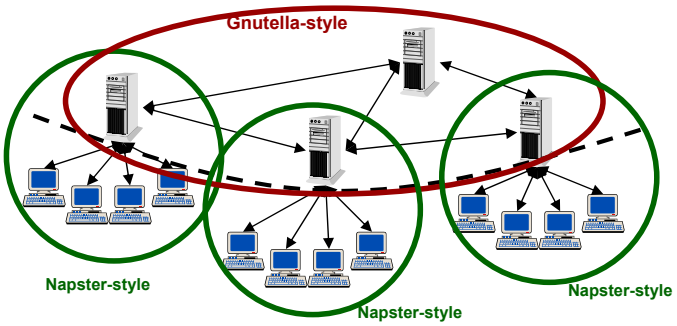
Η ανάκτηση πληροροφιών σε συστήματα ομοτίμων είναι μια περίπτωση κατανεμημένης ανάκτησης

Ιδιαιτερότητες των ομότιμων συστημάτων:

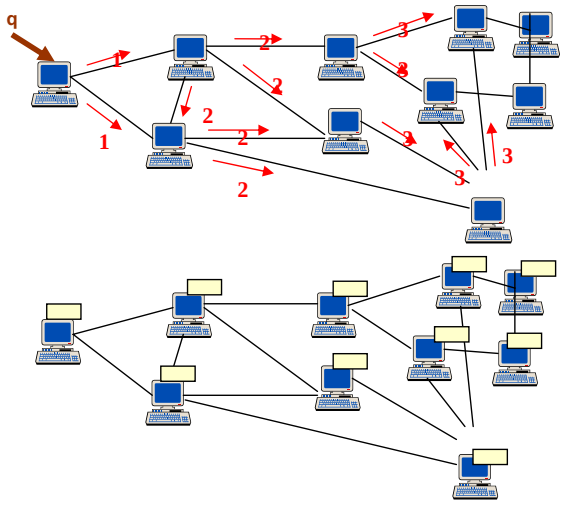
- Υπερβολικά μεγάλος αριθμός πηγών (peers)
- Μεγαλύτερη αυτονομία πηγών
- Έλλειψη Σταθερότητας, Ελέγχου, Προβλεψιμότητας
  - (not stable, controllable, unpredictable)
- Επιτακτική ανάγκη για μείωση του κόστους επικοινωνίας

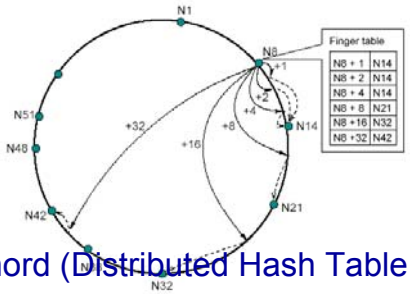
Αρχιτεκτονικές Ομότιμων Συστημάτων





CAN (Content Addressable Network)

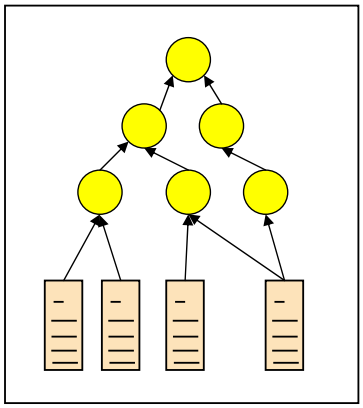




Chord (Distributed Hash Table -DHT)

P2P and IR: Περίπτωση: Κατηγοριοποιημένα Έγγραφα

Έστω ότι κάθε έγγραφο είναι ταξινομημένο σε μια κατηγορία ενός ελεγχόμενου ευρετηρίου (ODP, Yahoo!). Ο χρήστης κάνει αναζήτηση δίνοντας μια κατηγορία



έγγραφο ≈ mp3 αρχείο

κατηγορία εγγράφου ≈ τίτλος του mp3 αρχείου

Άρα μπορούμε να φτιάξουμε ένα ομότιμο σύστημα

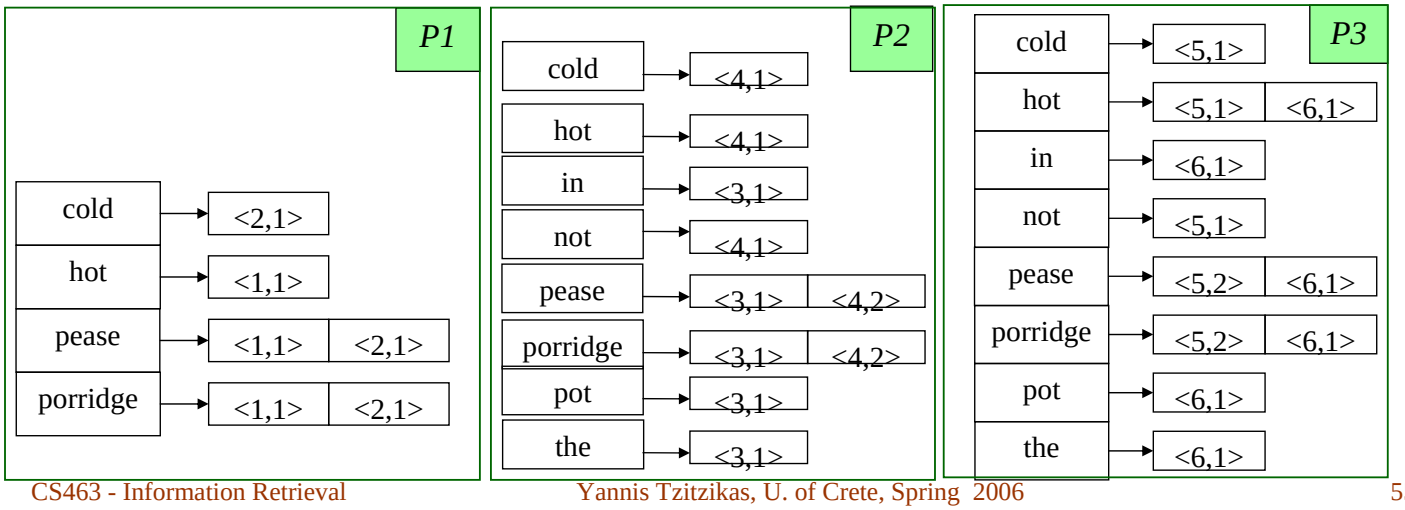
- τύπου Napster (Hybrid P2P)
- τύπου Gnutella (Pure P2P)
- τύπου Kazaa (Hierarchical P2P)
- τύπου Freenet (Structured P2P)
- τύπου Chord (Structured P2P)
- τύπου CAN (Structured P2P)

P2P and Statistical IR

Τυπικό Ευρετήριο P2P

p	k	d
121.111.86	”Singing in the Rain”	SR.mp3
121.111.86	«Υπάρχω»	stelios.mp3
222.18.78	“Singing in the Rain”	SingRain.mp3

Τυπικό Ευρετήριο IRS
(document partitioning)

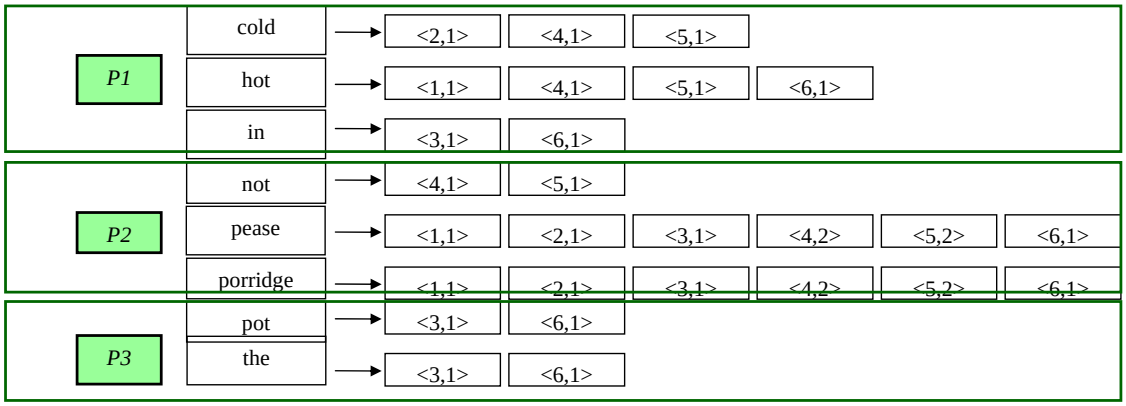


P2P and Statistical IR

Τυπικό Ευρετήριο P2P

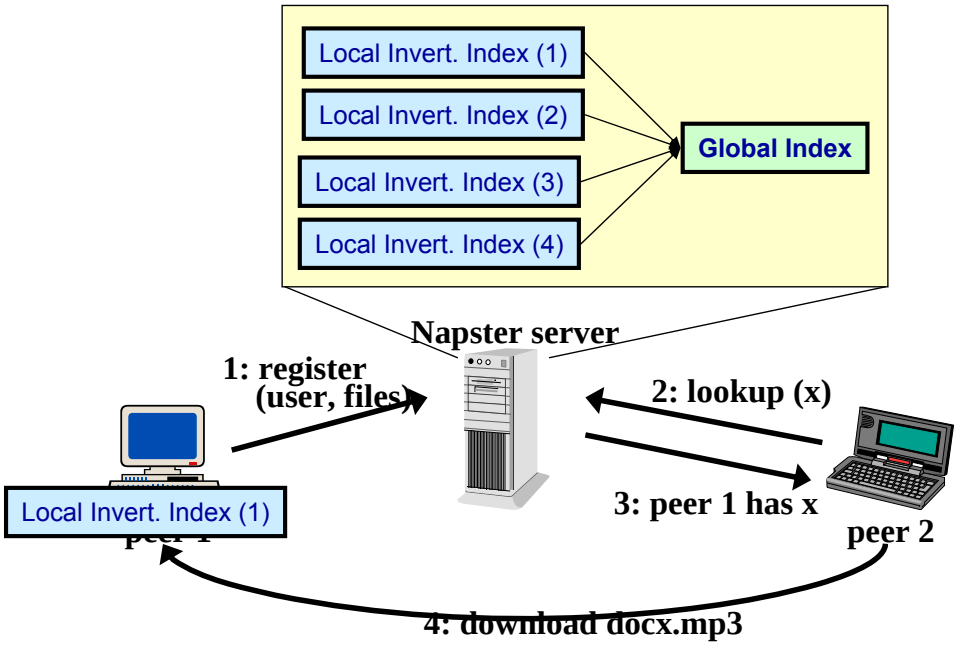
p	k	d
121.111.86	”Singing in the Rain”	SR.mp3
121.111.86	«Υπάρχω»	stelios.mp3
222.18.78	“Singing in the Rain”	SingRain.mp3

Τυπικό Ευρετήριο IRS
(term partitioning)



P2P and IR: Ανεστραμμένα Ευρετήρια (Napster-style)

- Ένας κεντρικός εξυπηρετητής αποθηκεύει όλα τα ανεστραμμένα ευρετήρια των κόμβων

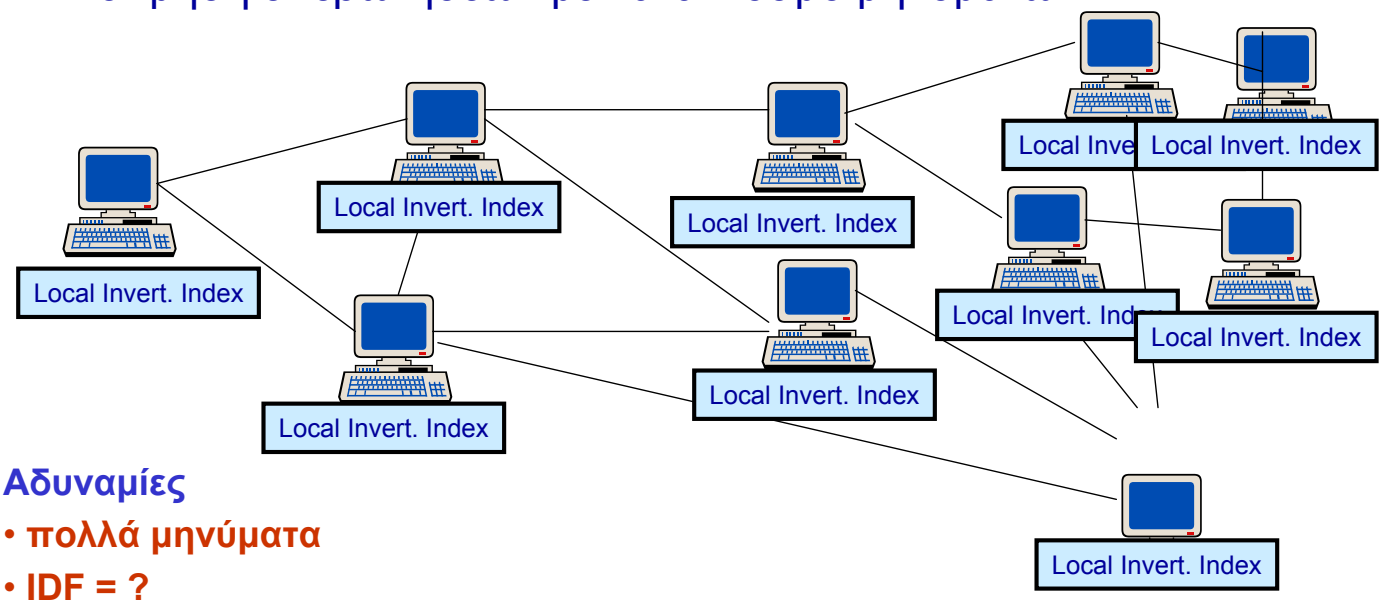


αδυναμίες:

- ο εξυπηρετητής χρειάζεται πολύ χώρο
- χρονοβόρο upload των ευρετηρίων στον εξυπηρετητή,
- το κόστος αποτίμησης επερωτήσεων πάει εξ' ολοκλήρου στον εξυπηρετητή
- ≈ Google, χωρίς το crawling (συλλογή σελίδων) και έχοντας έτοιμα κομμάτια του ευρετηρίου

P2P and IR: (Gnutella-style)

- Κάθε κόμβος συντηρεί το ανεστραμμένο ευρετήριο των εγγράφων του.
- Αποτίμηση επερωτήσεων με κατακλυσμό μηνυμάτων



Αδυναμίες

- **πολλά μηνύματα**
- **IDF = ?**

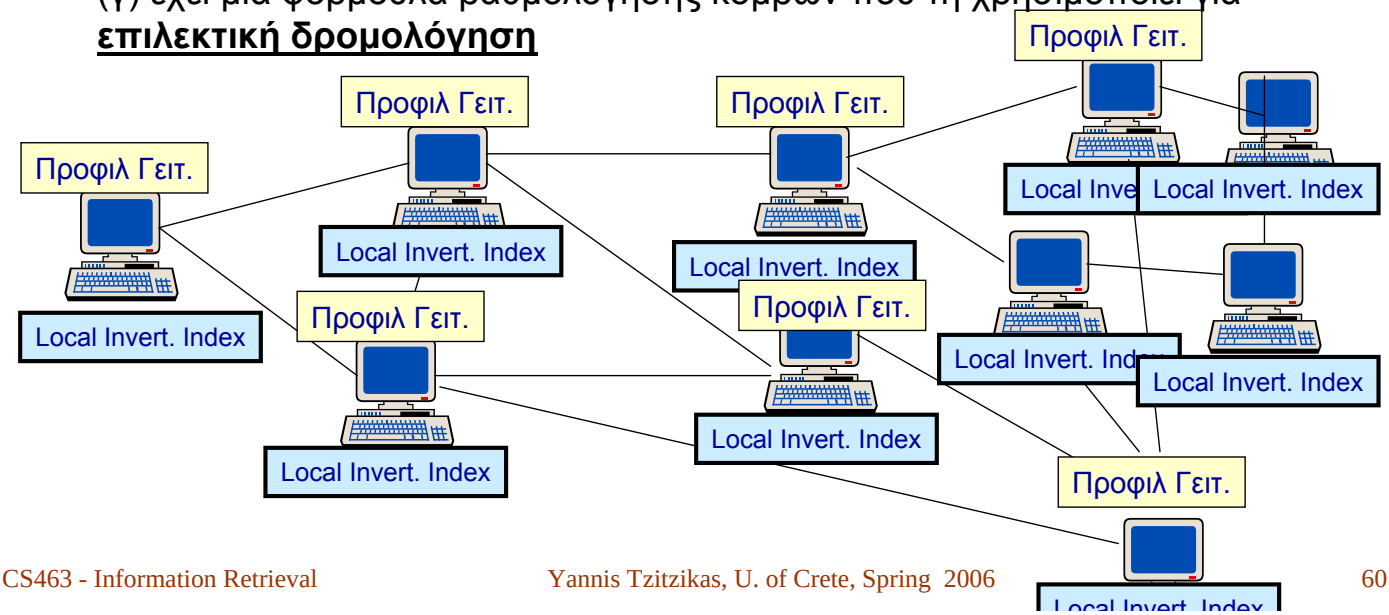
P2P and IR: (Gnutella-style)

Παραλλαγές του Κατακλυσμού μηνυμάτων

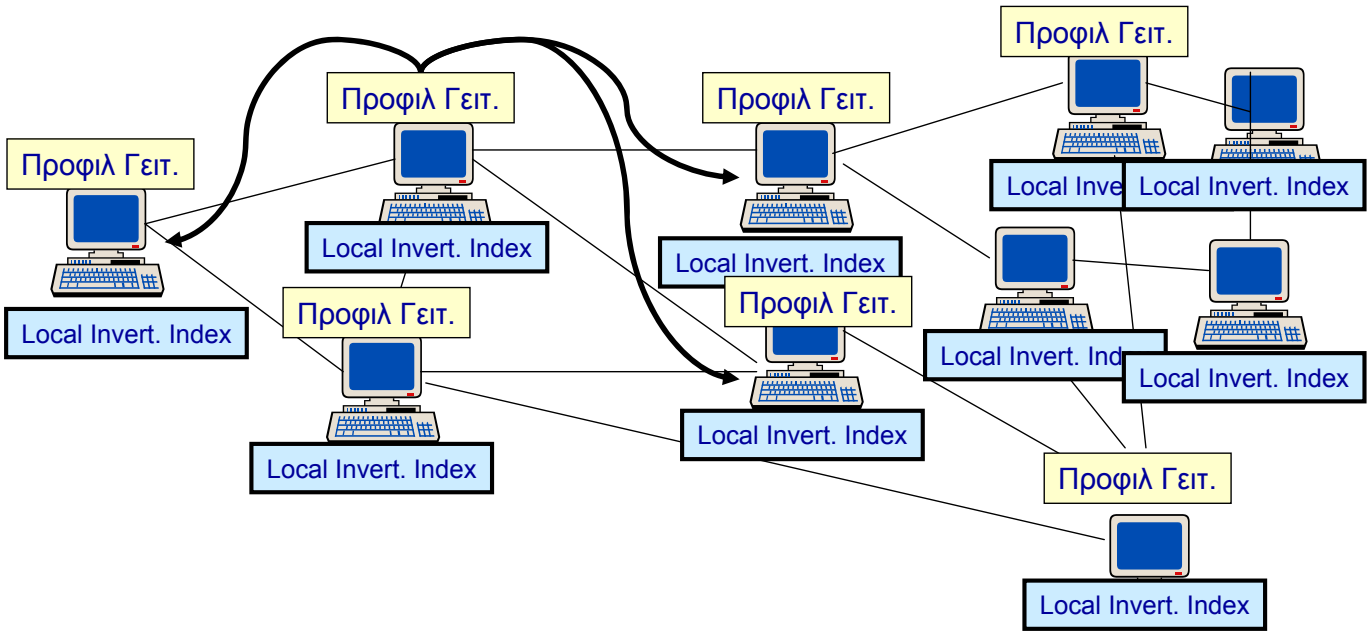
- BFS: Breadth First Search (=Gnutella)
- RBFS: κάθε κόμβος προωθεί ένα μήνυμα σε ένα τυχαίο ποσοστό (π.χ. 20%) των γνωστών του κόμβων
  - + πιθανοκρατικός αλγόριθμος
  - - μπορεί το μήνυμα να μην πάει σε κόμβους που έχουν συναφή αντικείμενα
- 1-Random Walker:
  - κάθε κόμβος προωθεί ένα μήνυμα σε <u>έναν τυχαία επιλεγμένο</u> κόμβο από τους γνωστούς του
- k-Random Walkers:
  - κάθε κόμβος προωθεί ένα μήνυμα σε <u>κ τυχαία επιλεγμένους</u> κόμβους από τους γνωστούς του
  - + λιγότερα μηνύματα από το RDFS
- APS: Adaptive Probabilistic Search

P2P and IR (Freenet-style)

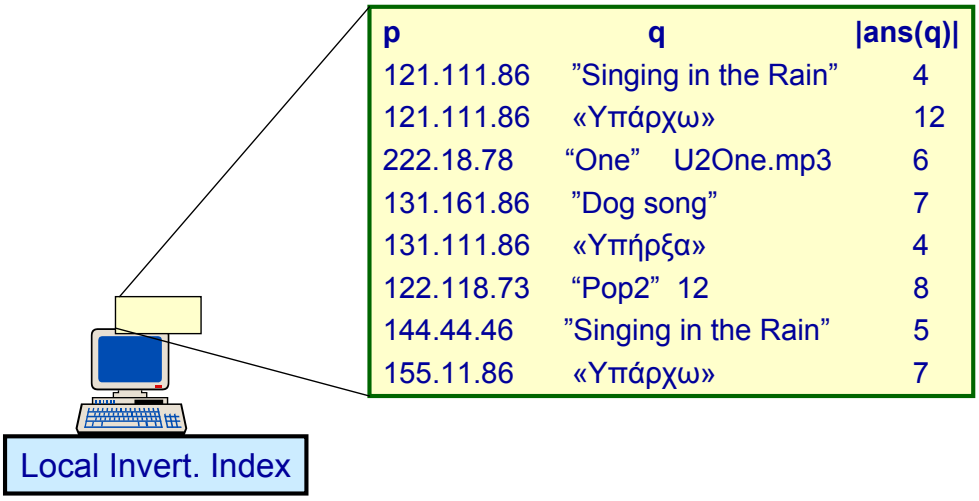
- Κάθε κόμβος:
  - (α) συντηρεί το ανεστραμμένο ευρετήριο των εγγράφων του.
  - (β) φτιάχνει ένα **<u>προφίλ</u>** <u>των γειτόνων</u> του βασισμένο στις επερωτήσεις του παρελθόντος
  - (γ) έχει μια φόρμουλα βαθμολόγησης κόμβων που τη χρησιμοποιεί για **<u>επιλεκτική δρομολόγηση</u>**



P2P and IR (Freenet-style)



Προφίλ Γειτόνων βάσει των προηγούμενων απαντήσεων

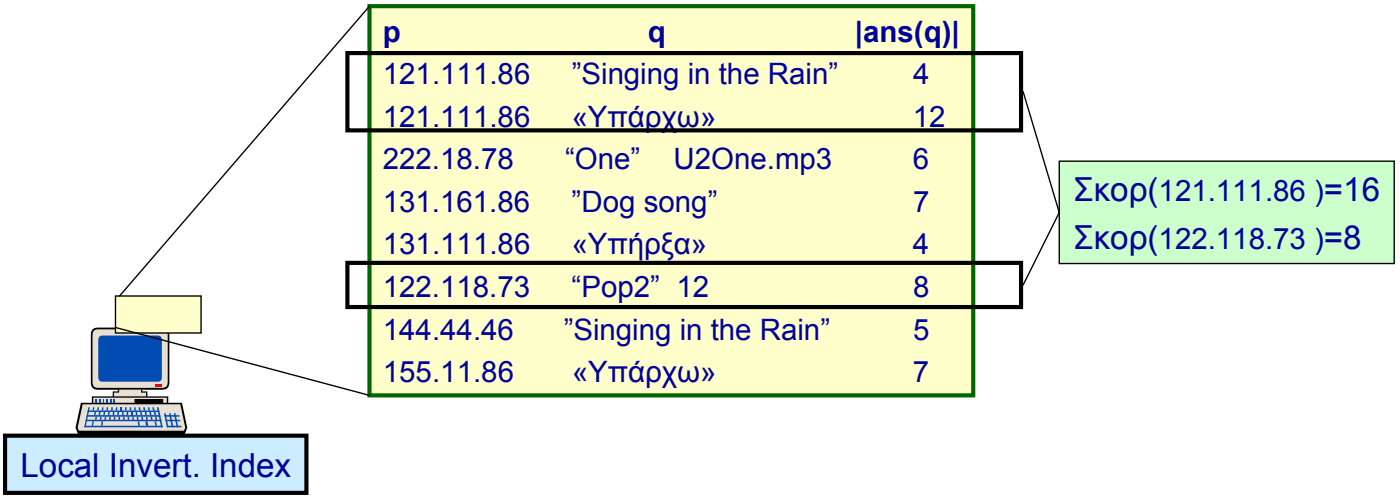


p	q	|ans(q)|
121.111.86	”Singing in the Rain”	4
121.111.86	«Υπάρχω»	12
222.18.78	“One” U2One.mp3	6
131.161.86	”Dog song”	7
131.111.86	«Υπήρξα»	4
122.118.73	“Pop2” 12	8
144.44.46	”Singing in the Rain”	5
155.11.86	«Υπάρχω»	7

LRU (Least Recently Used) deletion policy

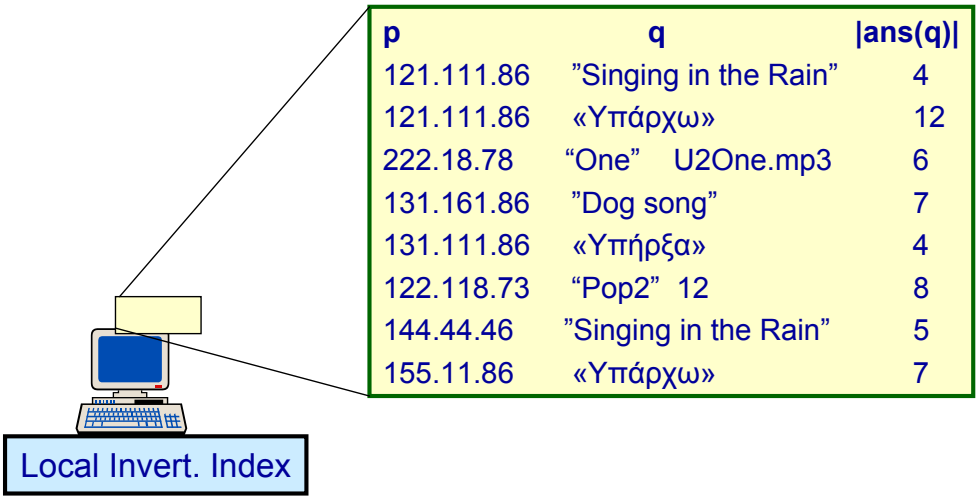
- Το **προφίλ** είναι τριάδες της μορφής (pj,q, |ans(pj,q)|)
  - όπου pj ένας γείτονας, q μια επερώτηση που απήντησε αυτός ο γείτονας, και |ans(pj,q)| το μέγεθος της απάντησης
  - LRU update policy

Προφίλ Γειτόνων και Δρομολόγηση:
>RES (*περισσότερα αποτελέσματα*)



- Για την δρομολόγηση μιας επερώτησης επιλέγονται εκείνοι οι γείτονες που έχει δώσει τα περισσότερα αποτελέσματα στο παρελθόν (>RES) (συγκεκριμένα στις προηγούμενες m επερωτήσεις)
- Το **σκορ ενός γείτονα pj είναι**
  - Score(pj) = Σ { |ans(pj,qj)| | qj answered by pj in the past}

Προφίλ Γειτόνων και Δρομολόγηση:
>RES και ομοιότητα επερωτήσεων



- Για την δρομολόγηση μιας επερώτησης q επιλέγονται εκείνοι οι γείτονες που έχουν δώσει τα περισσότερα αποτελέσματα στο παρελθόν (>RES) σε **επερωτήσεις που είναι κοντινές με το q**

Προφίλ Γειτόνων και Δρομολόγηση:
>RES και ομοιότητα επερωτήσεων

Το **σκορ ενός γείτονα pj δοθείσας επερώτησης q, είναι:**

>RES

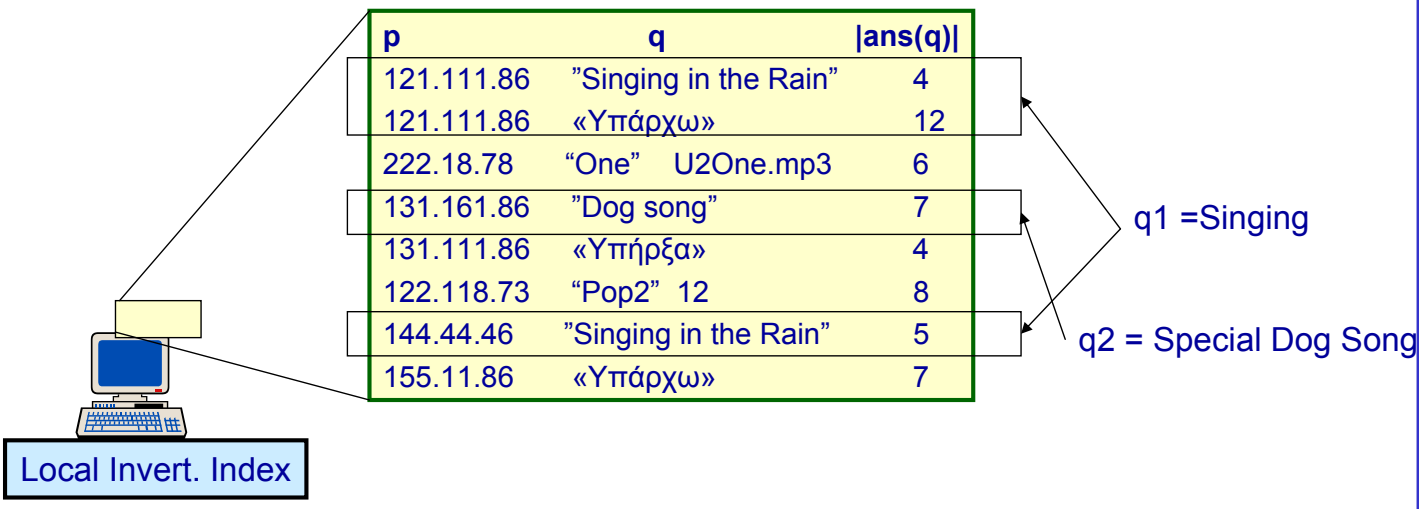
Score(pj) = Σ { |ans(pj,qj)| | qj answered by pj in the past}

>RES και **ομοιότητα επερωτήσεων**

Score(pj,**q**) = Σ { |ans(pj,qj)|*sim(qj, **q**)$^{\alpha}$ | qj answered by pj in the past}

- sim(qj, q): Π.χ. ομοιότητα συνημίτονου
- α: παράμετρος για το καθορισμό της σπουδαιότητας μεταξύ συνάφειας και μεγέθους απάντησης

Προφίλ Γειτόνων και Δρομολόγηση:
>RES και ομοιότητα επερωτήσεων

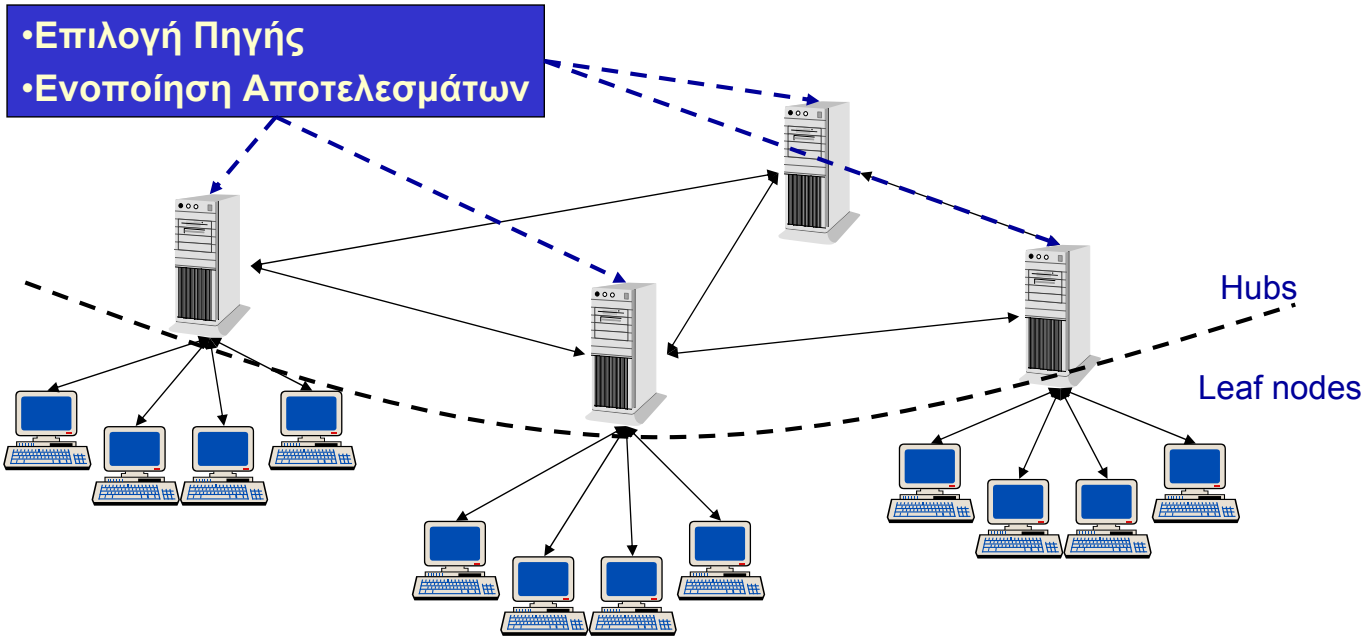


- Πότε αυτή η προσέγγιση είναι καλή;
- Απ: Όταν τα έγγραφα του κάθε κόμβου είναι σημασιολογικά κοντινά
  - Ποια η διαφορά με το Freenet ?
- Επειδή αυτό όμως δεν συμβαίνει πάντα η επερώτηση προωθείται **και** σε έναν **τυχαία** επιλεγμένο γείτονα. // επίσης για την εκκίνηση του συστήματος

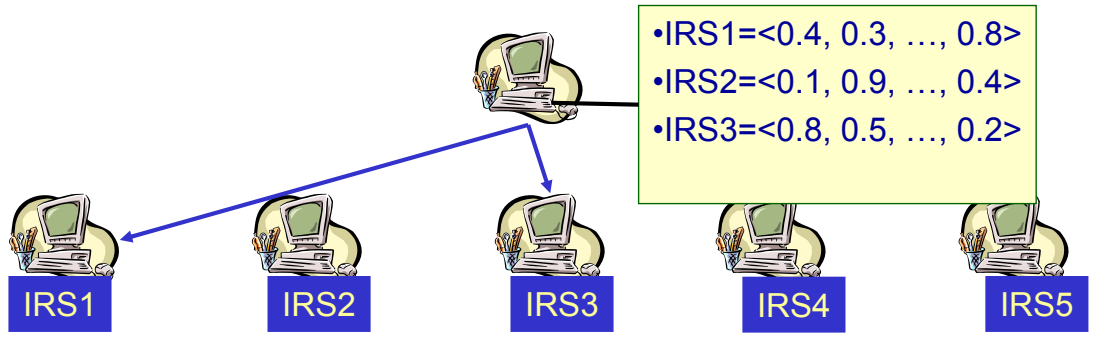
Ανάκτηση Κειμένων σε *Ιεραρχικά Ομότιμα Συστήματα* (Kazaa-style IR)

Γενική Ιδέα: Κάνουμε ό,τι και στην κατανεμημένη, απλά εδώ έχουμε πολλούς μεσίτες

Κάθε μεσίτης (εδώ super-peer) έχει μια περιγραφή των περιεχομένων των υποκείμενων κόμβων



Επανάληψη: Επιλογή Πηγής με **Διανύσματα Πηγών**

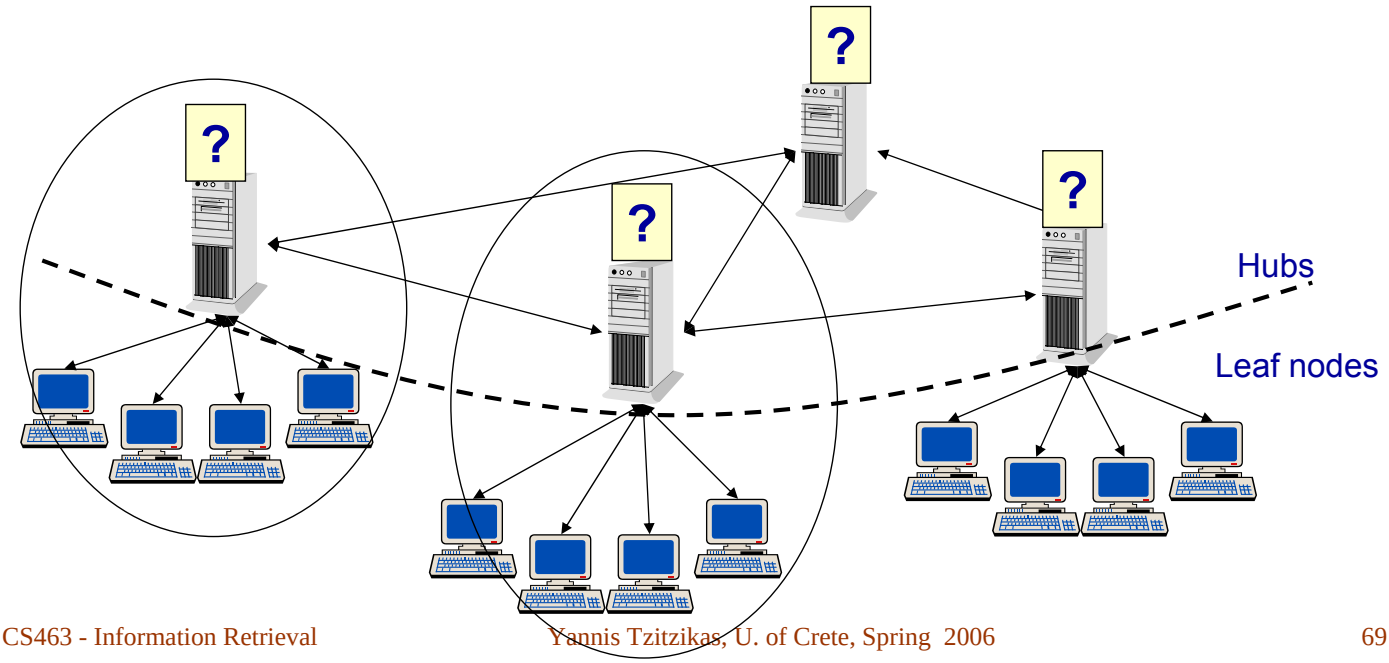


- Βλέπουμε **κάθε συλλογή** ως ένα **μεγάλο έγγραφο**
- Φτιάχνουμε ένα **διάνυσμα για κάθε συλλογή** (τύπου TF-IDF)
  - tfij: συνολικές εμφανίσεις του όρου i στη συλλογή j
  - idfi: log(N/ni), όπου N το πλήθος των συλλογών, και ni το πλήθος των συλλογών που έχουν τον όρο i
- Υπολογίζουμε το **βαθμό ομοιότητας κάθε νέας επερώτησης με το διάνυσμα κάθε συλλογής** (π.χ. ομοιότητα συνημίτονου)
- Διατάσσουμε τις συλλογές και επιλέγουμε τις κορυφαίες

Εναλλακτικά: Αντί για ένα, μπορούμε να περιγράφουμε κάθε πηγή με Κ διανύσματα

Περιγραφή των περιεχομένων των φύλλων

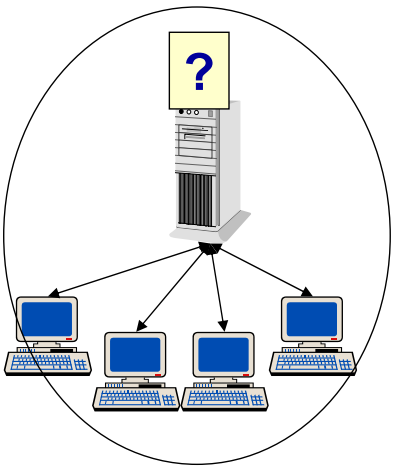
Ανάγκη για μείωση του αποθηκευτικού χώρου στα Hubs



Επιλογές

1/ Λεξιλόγια των υποκείμενων κόμβων + συχνότητες εμφάνισης τους

- (δεν ξέρουμε το καθολικό λεξιλόγιο για να φτιάξουμε το διάνυσμα πηγής)

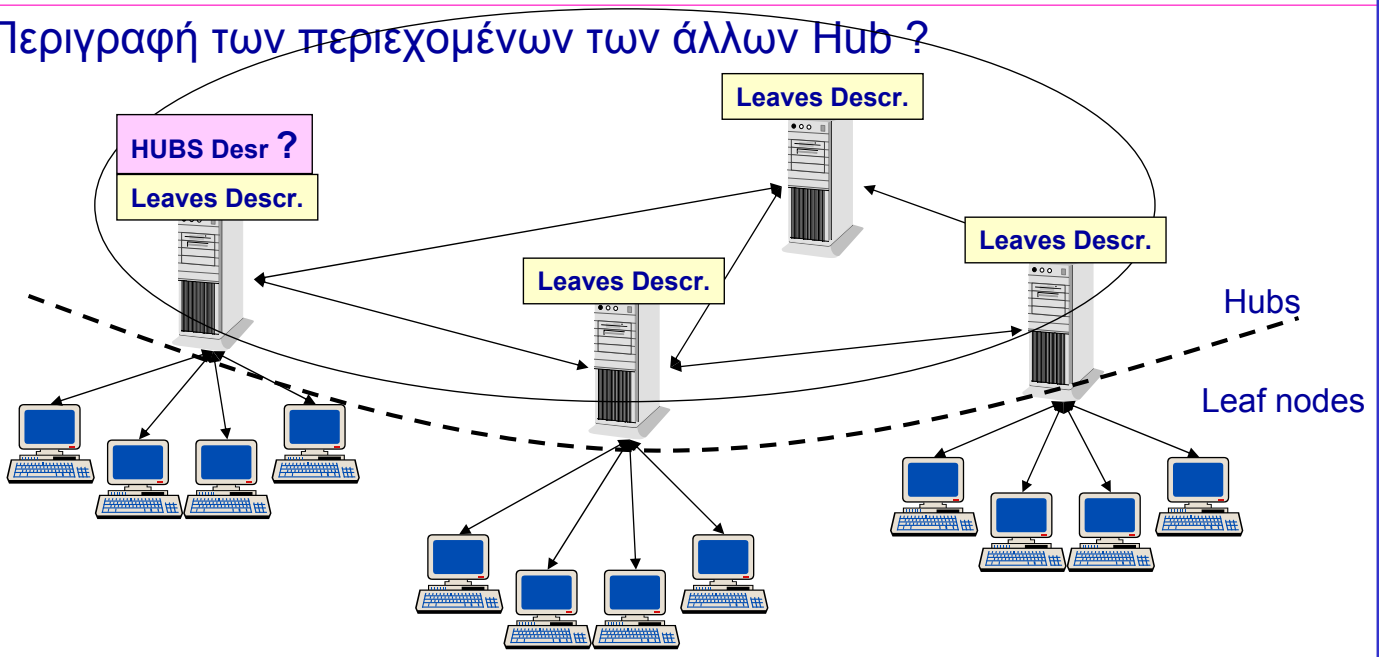


2/ Λεξιλόγια των υποκείμενων κόμβων

3/ Λέξεις που εμφανίζονται πάνω από 1 φορά + συχνότητες τους

- λόγω του νόμου του Ziph, ο απαιτούμενος αποθηκευτικός χώρος μειώνεται στο μισό

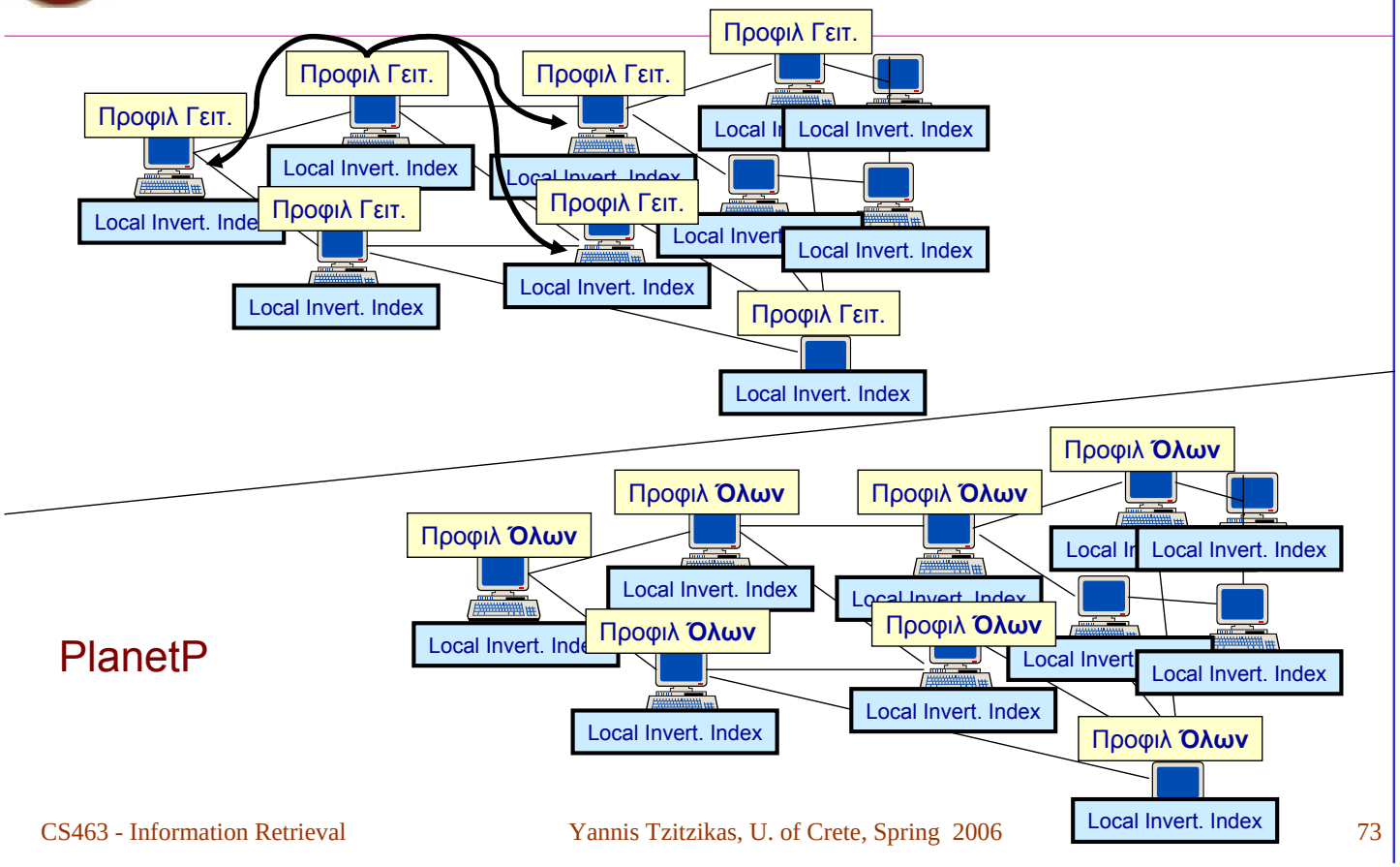
Περιγραφή των περιεχομένων των άλλων Hub ?



- Η περιγραφή ενός HUB είναι η ένωση των περιγραφών των υποκείμενων του κόμβων (Πρόβλημα: χώρος)
- Καταγραφή προηγούμενων επερωτήσεων που έχουν απαντηθεί
  - π.χ. >RES και ομοιότητα επερώτησης

- A Client node sends its query to each of its connecting hubs.
- A hub that receives the query uses its **resource selection algorithm** to rank and select one or more neighboring leaf nodes as well as hubs, and routes the query to them if the message's TTL hasn't reached 0.
- A leaf node that receives the query message uses its **document retrieval algorithm** to generate a relevance ranking of its documents and responds with a queryhit message to include a list of top-ranked documents.
- Each top-level hub (the hub that connects directly to the client node that issues the request) collects the queryhit messages and uses its **result merging algorithm** to merge the documents retrieved from multiple leaf nodes into a single, integrated ranked list and returns it to the client node.
- If the client node issues the request to more than one hub, then it also needs to merge results returned by multiple toplevel hubs.

P2P and IR: Το σύστημα **PlanetP**

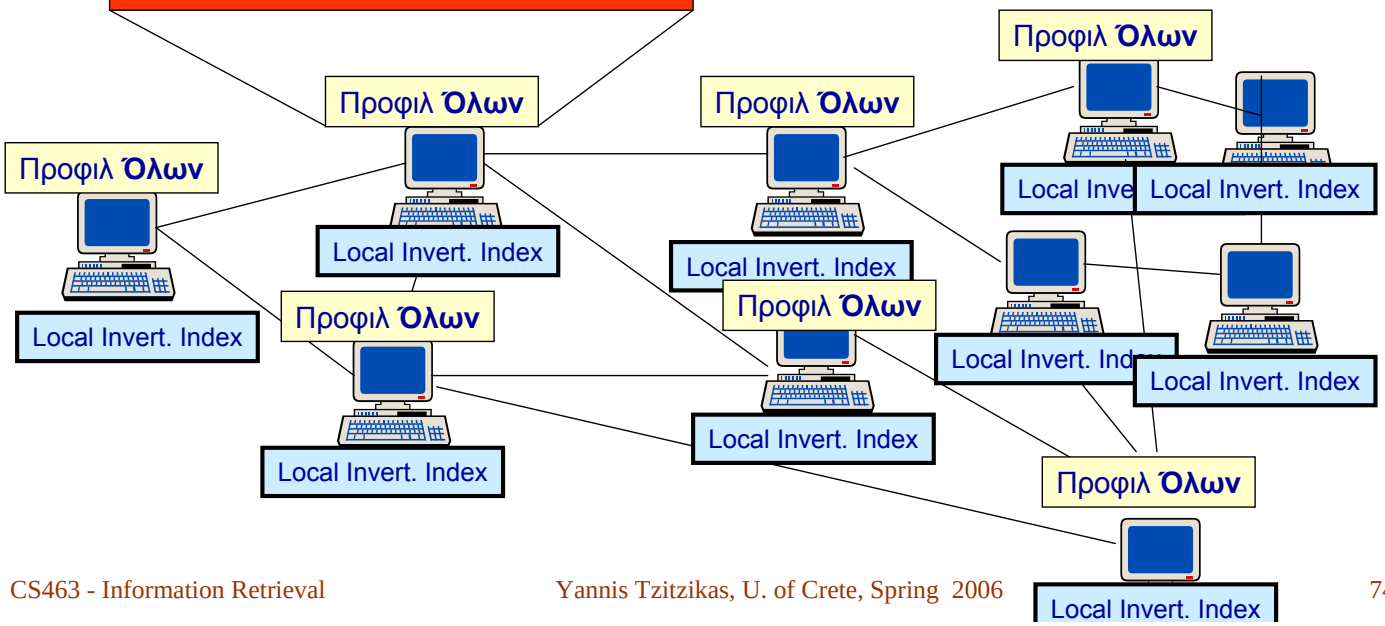


P2P and IR: Το σύστημα **PlanetP**

καθολικό
ανεστραμμένο ευρετήριο όλων
των κόμβων

? ΟΧΙ.

Ανάγκη για μια **πιο συνοπτική περιγραφή**



P2P and IR: Το σύστημα **PlanetP**
Επιλογή Πηγής ???

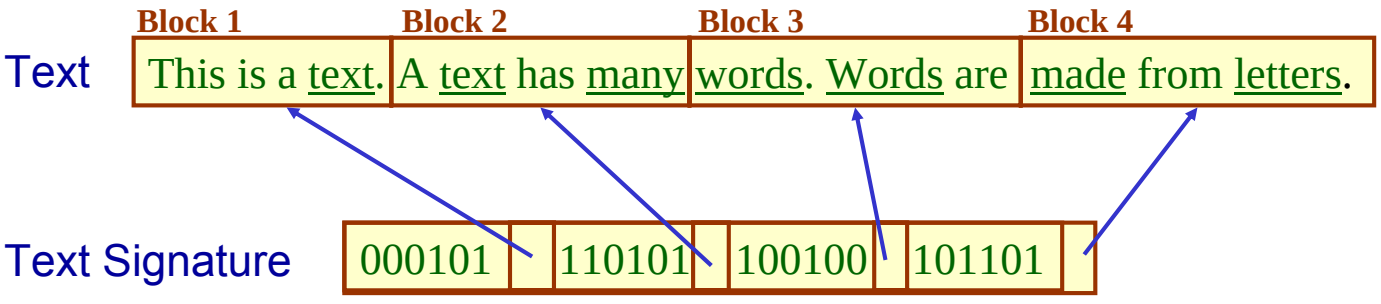
Θυμηθείτε:
Κατανομή Συναφών Εγγράφων
(Relevant document distribution (RDD))
Διανύσματα Πηγών

- Το **λεξιλόγιο κάθε κόμβου** (όχι οι λίστες των εμφανίσεων) περιγράφεται με ένα **Bloom φίλτρο**

- ~ SIGNATURE FILES

Επανάληψη: Signature files

b=3 (3 words per block) B=6 (bit masks of 6 bits)



Signature Function

h(text)= 000101
h(many)= 110000
h(words)=100100
h(made)= 001100
h(letters)=100001

Bloom filters [Burton Bloom 1970]
Συμπαγής Κωδικοποίηση Συνόλων

Ένα σύνολο κωδικοποιείται σε ένα δυαδικό διάνυσμα με των **m**-bits

κ συναρτήσεις κατακερματισμού h1, h2, … , hk, με πεδίο τιμών το {1,..,m}

Κωδικοποίηση στοιχείου:

BF({α}) = διάνυσμα με άσσους στις θέσεις h1(α), h2(α), … , hk(α)

Κωδικοποίηση συνόλου:

BF({α1, α2}) = BF({α1}) BITwiseOR BF({α2})

Πως βρίσκω αν ένα στοιχείο b ανήκει στο σύνολο A ?

1/ Υπολογίζω το BloomFilter του b

2/ Κοιτάζω αν οι άσσοι του BF(b) υπάρχουν στο BF(A)

Αν όχι, τότε σίγουρα το b δεν ανήκει στο A

Αν ναι, τότε ανήκει αλλά μπορεί και να μην ανήκει (false positive)

Όσο μεγαλύτερο είναι το m, τόσο μικρότερη η πιθανότητα για false positives

Bloom filters: Παράδειγμα

m=14, k=3 

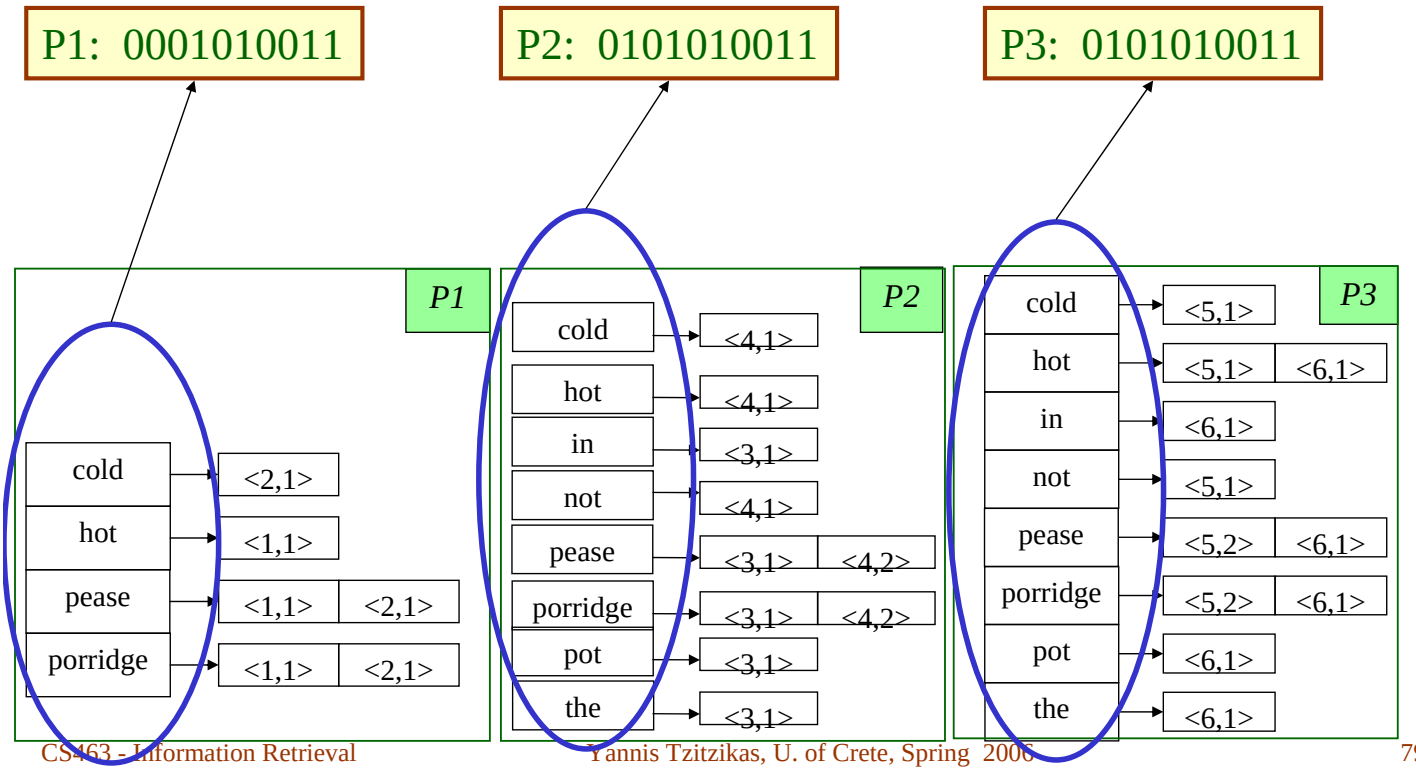
hash1("apples") = 3
hash2("apples") = 12
hash3("apples") = 11

{apples}= 

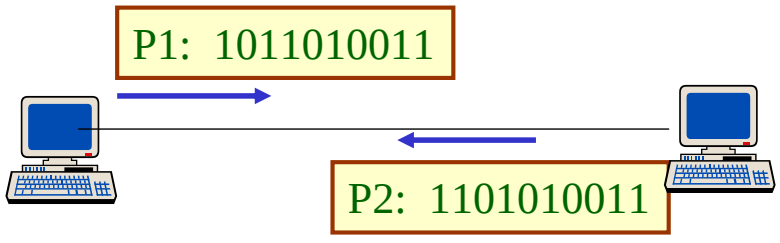
hash1("plums") = 11
hash2("plums") = 1
hash3("plums") = 8

{apples, plums}= 

Περιγραφή των λεξιλογίων με Bloom filters



P2P and IR: Το σύστημα **PlanetP**



Κάθε pi κατασκευάζει το bloom filter bi του δικού του λεξιλογίου και στο στέλνει στο υπόλοιπο δίκτυο με τεχνικές flooding/gossiping

Κάθε pi ξέρει τα bloom filters όλων των άλλων κόμβων

Έτσι κάθε pi μπορεί να βρεί τους κόμβους που έχουν έναν συγκεκριμένο όρο (άρα μπορεί να προσεγγίσει το καθολικό ευρετήριο)

Bloom filters in PlanetP: Πόσο μεγάλα είναι;

AP89 Collection (Associated Press articles of 1989 from TREC):
84,678 documents, 129,603 words, collection size 266 MB

Num. Peers	Memory used (MB)	% of collection size
10	0.45	0.18%
100	1.79	0.70%
1000	4.48	1.76%

Γιατί το μέγεθος αυξάνει με το πλήθος των κόμβων;

1000 Nodes: => about 4500 terms per peer
Bloom filters with less than 5% false positives =>
Bloom filter size for the vocabulary of one peer: 4.6 KB
Total size of bloom filters of peers : 4.6 MBytes

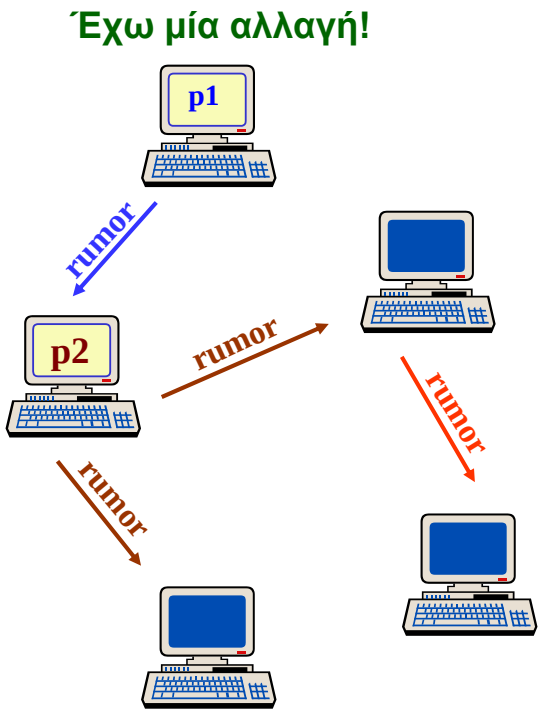
PlanetP: Τρόπος ενημέρωσης των κόμβων (Gossiping algorithms)

- Η μετάδοση των Bloom filters σε όλο το δίκτυο καθώς και η ενημέρωση των κόμβων (για νέα δεδομένα, είσοδο/έξοδο κόμβων) μπορεί να γίνει με ποικίλους **αλγορίθμους gossiping**:
  - rumoring algorithm
  - anti-entropy algorithm
  - partial anti-entropy algorithm
  - ….

(Gossiping algorithms)
Rumoring (φημολογία)

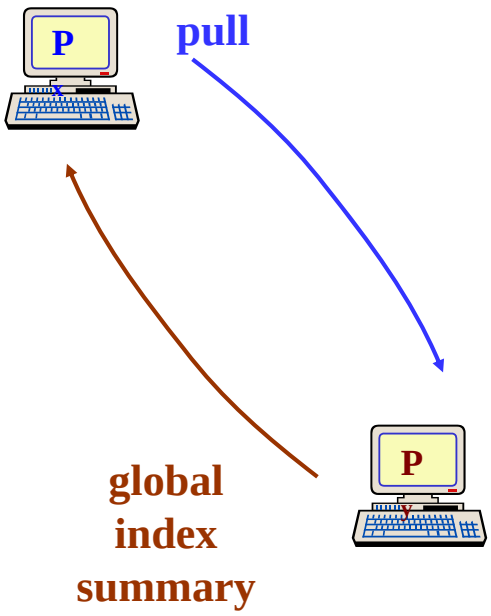
Ο p1 έχει μια αλλαγή:

- κάθε Χ δευτερόλεπτα, ο p1 στέλνει ένα μήνυμα με την αλλαγή σε έναν τυχαία επιλεγμένο κόμβο p2
- Αν ο p2 δεν ήξερε αυτήν την πληροφορία. τότε αρχίζει να κάνει ό,τι και ο p1
- Ο p1 σταματάει να στέλνει μηνύματα μόνο αν η συνεχόμενοι κόμβοι του πουν ότι ήταν ήδη ενήμεροι της αλλαγής.

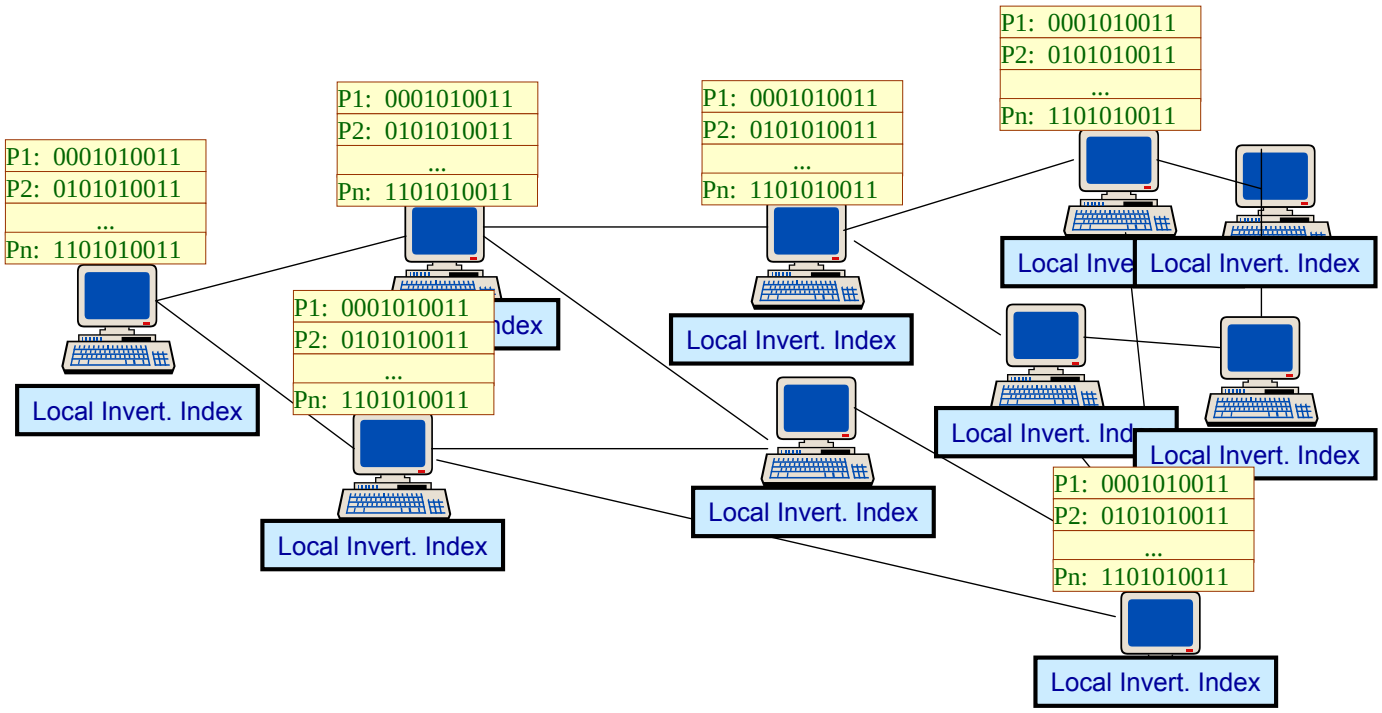


(Gossiping algorithms)
anti-entropy

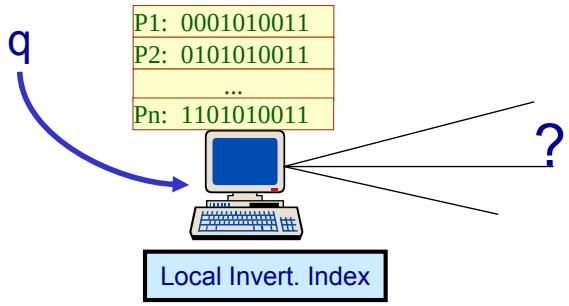
- Κάθε Χ δευτερόλεπτα, **κάθε κόμβος** επιλέγει τυχαία έναν άλλο κόμβο (από το καθολικό του ευρετήριο) και του ζητάει να του στείλει μια περίληψη το δικού του καθολικού ευρετήριο.
- Αν διαπιστώσει ότι δεν είναι ενημερωμένος, του ζητάει ό,τι χρειάζεται.
- Purpose: The algorithm allows to avoid the possibility of rumors dying out before reaching everyone



P2P and IR: Το σύστημα **PlanetP**



PlanetP: Επιλογή Κόμβου



1/ **Βαθμολόγηση κόμβων** βάσει της πιθανότητας να έχουν έγγραφα συναφή με την q

2/ **Επιλογή** των κόμβων που θα επερωτηθούν και ενοποίηση των αποτελεσμάτων που θα επιστρέψουν

Inverse Peer Frequency (IPF) of a term t =

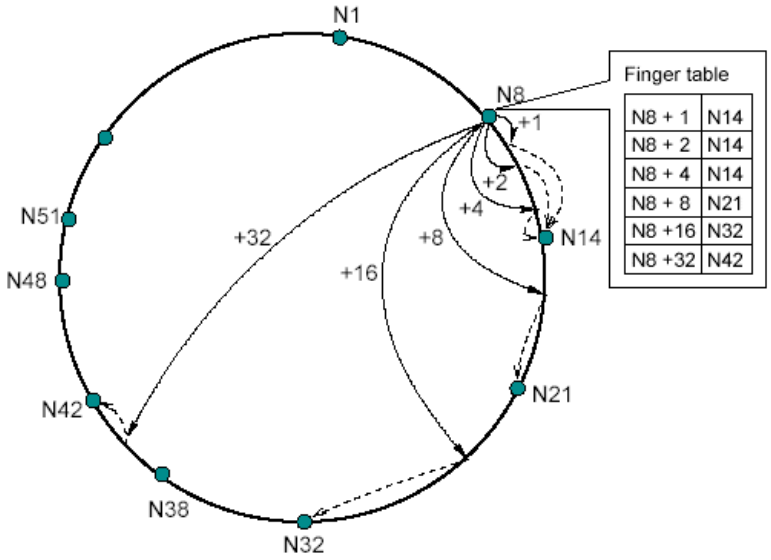
IPF(t) := |total number of peers| / |peers that contain the term t|

$$Score(pj,\mathbf{q}) = \Sigma \{ IPF(t) \mid t \in q,\ t \in Bfilter(pj) \}$$

PlanetP: Αποτελεσματικότητα & Επιδόσεις

- Η αποτελεσματικότητα προσεγγίζει αυτήν που θα είχαμε αν κάθε κόμβος είχε ολόκληρο το ευρετήριο
- Τα μηνύματα φτάνουν σε 20%-40% περισσότερους κόμβους σε σχέση με την περίπτωση όπου κάθε κόμβος γνώριζε ακριβώς το καθολικό ευρετήριο
- Gossiping rate 1/second => PlanetP can propagate a Bloom filter containing 1000 terms in less than 40 secs for a community of 1000 peers. This requires an average of 24KB/s per peer.

Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (Chord-style)

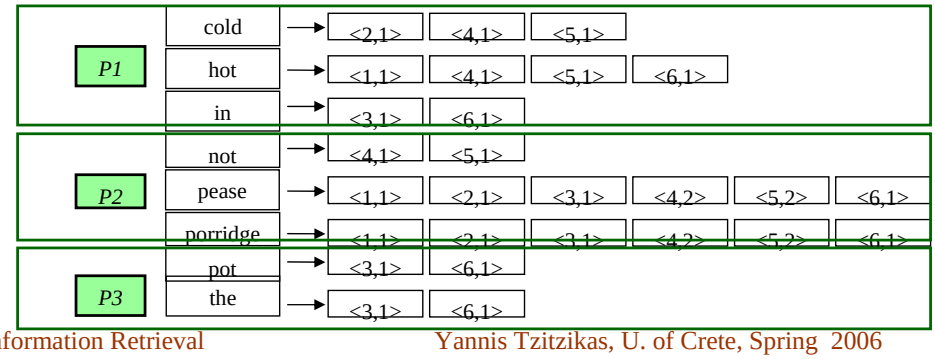


- Ποια είναι εδώ τα κλειδιά ?

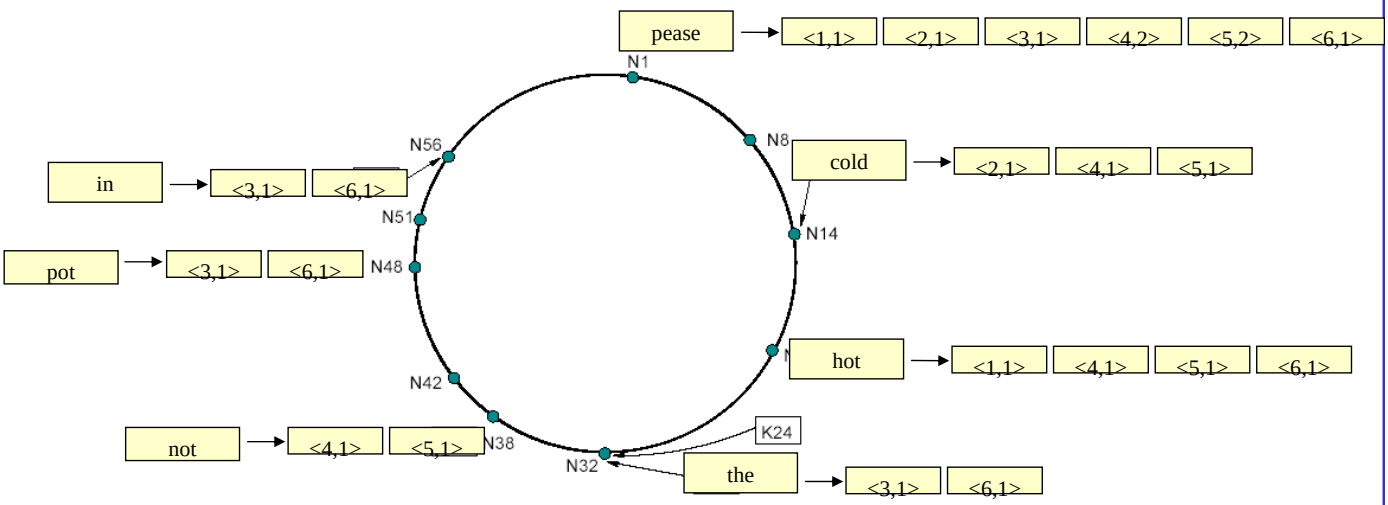
Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (Chord-style)

Περίπτωση (I): Κάθε όρος είναι ένα κλειδί

- Το ευρετήριο κατανέμεται βάσει των όρων
  - (άρα έχουμε term-partitioning: θυμηθείτε την παράλληλη Α.Π.)
- Αδυναμία: Η ενημέρωση των ευρετηρίων είναι ακριβή:
  - Εισαγωγή ενός νέου εγγράφου:
    - Για κάθε λέξη του εγγράφου, πρέπει να βρούμε τον κόμβο που είναι υπεύθυνος για αυτήν την λέξη και να του στείλουμε την ανεστραμμένη λίστα

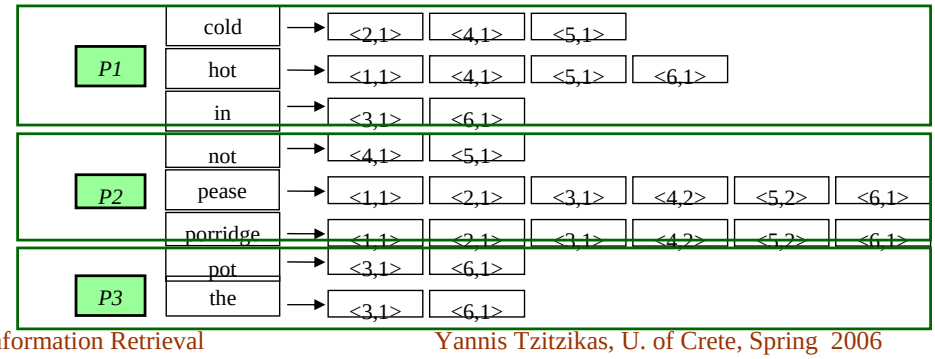


Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (Chord-style): Κάθε όρος είναι ένα κλειδί



Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (Chord-style)

- Αποτίμηση επερώτησης q
  - βρίσκουμε κάθε κόμβο που έχει τουλάχιστον έναν όρο του q (χρησιμοποιώντας τους πίνακες δρομολόγησης)
  - Σενάριο 1: κάθε ένας από αυτούς τους κόμβος υπολογίζει τα μερικά σκορ και τα στέλνει στον ερωτώντα (αφού του στείλουμε και την επερώτηση)
  - Σενάριο 2: κάθε ένας από αυτούς τους κόμβους επιστρέφει τις ανεστραμμένες λίστες
  - [-] Ανταλλαγή πολλών μηνυμάτων για επερωτήσεις με πολλούς όρους



Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (Chord-style)

Υπόθεση: Έστω ότι το σύστημα λαμβάνει πολύ συχνά επερωτήσεις με 2 όρους

Περίπτωση (II): Θεωρούμε ως κλειδί κάθε **ζευγάρι** όρων

- Αν η επερώτηση έχει 2 όρους, τότε **ένας μόνο** κόμβος θα έχει όλο το κομμάτι του ευρετηρίου που χρειαζόμαστε
- Άρα έτσι έχουμε λίγα μηνύματα

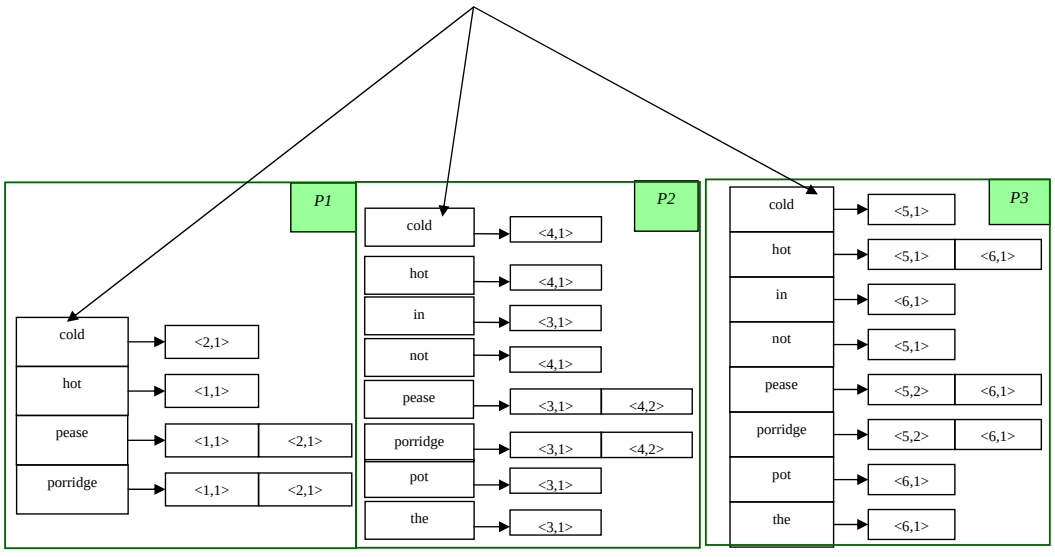
- Π.χ. q= Hotels Crete
  - Ξέρω ότι υπάρχει ένας κόμβος που έχει τις ανεστραμμένες λίστες και των δυο όρων, άρα ο κόμβος αυτός μπορεί να αποτιμήσει πλήρως την επερώτηση
- Αδυναμία: |V|* (|V|-1) κλειδιά, άρα η ανεστραμμένη λίστα κάθε λέξης είναι αποθηκευμένη |V|-1 φορές

P1	Hotels [….*inverted list for Hotels* ….] Crete [*...inverted list for Crete ...*]
P1	Hotels [….*inverted list for Hotels* ….] Cefalonia [*...inverted list for Cefalonia ...*]
P3	Crete [….*inverted list for Crete* ….] Cefalonia [*...inverted list for Cefalonia ...*]

Η είσοδος ενός νέου εγγράφου είναι ακόμα πιο ακριβή

- Περίπτωση (III): Θεωρούμε ως κλειδιά τα **διανύσματα** των **εγγράφων**
  - **Ερ**: Ποια προσέγγιση δομημένων συστημάτων είναι κατάλληλη για την παράσταση διανυσμάτων (Chord ή CAN) ;
  - **Απ**: Η προσέγγιση του CAN διότι βλέπει τον χώρο των κλειδιών ως ένα κ-διάστατο χώρο
  - Άρα διαμερίζουμε τα έγγραφα στους κόμβους βάσει των διανυσμάτων τους.
    - (άρα document-partitioning (θυμηθείτε την Παράλληλη Α.Π.))
  - **Ερ**: Τι κερδίζουμε διαμερίζοντας τα έγγραφα όπως το CAN ?
  - **Απ**: Τα κοντινά (ως προς το μέτρο συνημίτονου) έγγραφα τοποθετούνται στον ίδιο ή σε κοντινούς κόμβους.

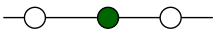
Document Partitioning: Ο υπολογισμός των καθολικών στατιστικών (IDF) απαιτεί επικοινωνία

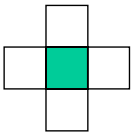


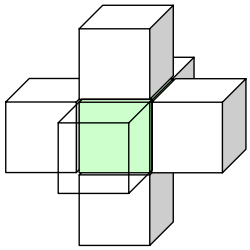
Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (**CAN**-style)

- **Ερ**: Πόσες διαστάσεις έχουν τα διανύσματα των εγγράφων;
- **Απ**: Συνήθως πολλές (π.χ. 10.000)

- **Ερ**: Πόσους γείτονες έχει μια περιοχή k-διάστατου χώρου;
- **Απ**: κατά μέσο όρο 2κ
  - Για κ=1 έχω 2
  - Για κ=2 έχω 4
  - Για κ=3 έχω 6
  - Για κ=10.000 έχω 20.000 !





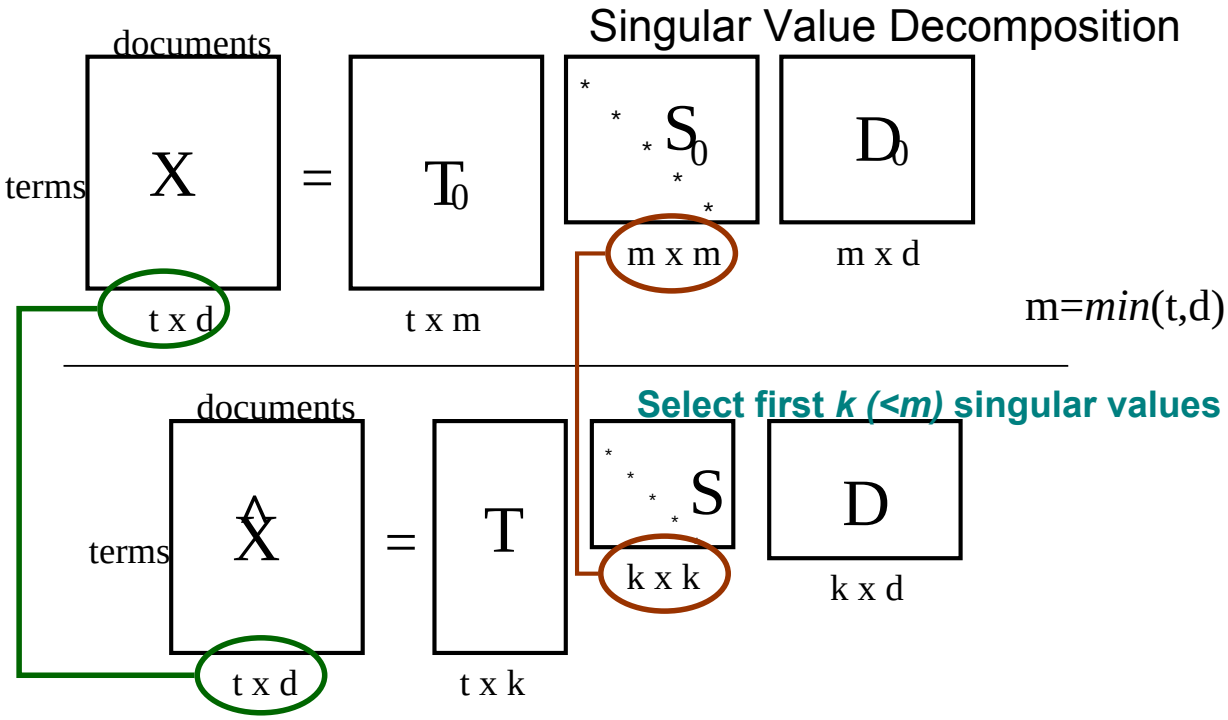


Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (CAN-style). Το σύστημα **pSearch**

- Μείωση των διαστάσεων των διανυσμάτων για
  - (I) Μείωση του αριθμού των γειτόνων που πρέπει να γνωρίζει (αποθηκεύει) ένας κόμβος.
  - (II) Ομαδοποίηση εγγράφων
    - Αξιοποίηση συνωνύμων, συνεμφανιζόμενων λέξέων, μείωση θορύβου
- Τρόπος μείωσης διαστάσεων: **Latent Semantic Indexing**

Επανάληψη: Latent Semantic Indexing:

t: total number of index terms
d: total number of documents



Επανάληψη LSI: Paper example

Index terms in italics

Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Επανάληψη LSI: term-document Matrix

Terms	Documents								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Επανάληψη LSI: T_0,S_0 ,D_0

T_0

$$\begin{bmatrix}
0.22 & -0.11 & 0.29 & -0.41 & -0.11 & -0.34 & 0.52 & -0.06 & -0.41 \\
0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11 \\
0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49 \\
0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01 \\
0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27 \\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\
0.30 & -0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17 \\
0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58 \\
0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23 \\
0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.23 \\
0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18
\end{bmatrix}$$

S_0

$$\begin{bmatrix}
3.34 & & & & & & & & \\
& 2.54 & & & & & & & \\
& & 2.35 & & & & & & \\
& & & 1.64 & & & & & \\
& & & & 1.50 & & & & \\
& & & & & 1.31 & & & \\
& & & & & & 0.85 & & \\
& & & & & & & 0.56 & \\
& & & & & & & & 0.36
\end{bmatrix}$$

D_0

$$\begin{bmatrix}
0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06 \\
0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24 \\
0.46 & -0.13 & 0.21 & 0.04 & 0.38 & 0.72 & -0.24 & 0.01 & 0.02 \\
0.54 & -0.23 & 0.57 & 0.27 & -0.21 & -0.37 & 0.26 & -0.02 & -0.08 \\
0.28 & 0.11 & -0.51 & 0.15 & 0.33 & 0.03 & 0.67 & -0.06 & -0.26 \\
0.00 & 0.19 & 0.10 & 0.02 & 0.39 & -0.30 & -0.34 & 0.45 & -0.62 \\
0.01 & 0.44 & 0.19 & 0.02 & 0.35 & -0.21 & -0.15 & -0.76 & 0.02 \\
0.02 & 0.62 & 0.25 & 0.01 & 0.15 & 0.00 & 0.25 & 0.45 & 0.52 \\
0.08 & 0.53 & 0.08 & -0.03 & -0.60 & 0.36 & 0.04 & -0.07 & -0.45
\end{bmatrix}$$

Επανάληψη LSI: SVD with minor terms dropped

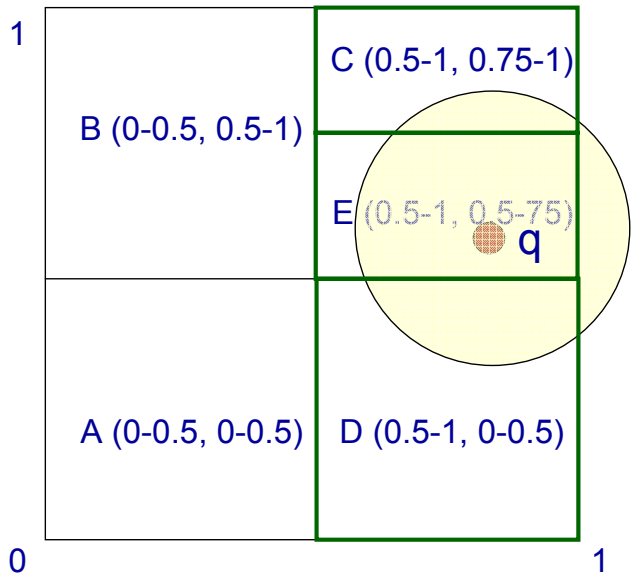
$$T = \begin{bmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{bmatrix} \quad S = \begin{bmatrix} 3.34 & \\ & 2.54 \end{bmatrix} \quad D' = \begin{bmatrix} 0.20 & 0.61 & 0.46 & 0.54 & 0.28 & 0.00 & 0.02 & 0.02 & 0.08 \\ -0.06 & 0.17 & -0.13 & -0.23 & 0.11 & 0.19 & 0.44 & 0.62 & 0.53 \end{bmatrix}$$

TS define coordinates for documents in latent space

Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (CAN-style)**.** Το σύστημα **pSearch**

- Διάσταση διανυσμάτων μετά την εφαρμογή LSI: **50-350**
- Φτιάχνουμε ένα CAN με διαστάσεις όσες των διανυσμάτων (μετά το LSI).
- Εισαγωγή ενός νέου εγγράφου:
  - Φτιάχνεται το «semantic διάνυσμα» του εγγράφου (βάσει των διαστάσεων που προέκυψαν από την εφαρμογή του LSI) και εισάγεται στον κατάλληλο κόμβο
- Είσοδος μιας νέας επερώτησης
  - Φτιάχνεται το semantic διάνυσμα της επερώτησης και δρομολογείται στον κατάλληλο κόμβο
  - Μόλις φτάσει στον κόμβο, διαδίδεται στους γείτονες σε απόσταση ρ
    - Το ρ μπορεί να δίδεται μαζί με την αρχική επερώτηση

- Επερώτηση

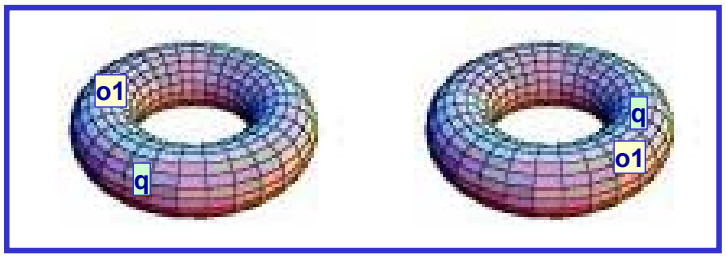


- Ο υπολογισμός του LSI απαιτεί Καθολικά στατιστικά (IDF)
- Επίσης όλοι οι κόμβοι πρέπει να γνωρίζουν την βάση του σημασιολογικού χώρου (για να υπολογίζουμε τα σημασιολογικά διανύσματα των νέων εγγράφων).
- Τα παραπάνω πρέπει να διαδοθούν σε όλους τους κόμβους.

- Το πρόβλημα των διαστάσεων
  - 300 LSI διαστάσεις. Αν έχω λίγους κόμβους τότε η πραγματική διάσταση του CAN ειναι μικρότερη γιατί δεν υπάρχουν αρκετοί κόμβοι. Έτσι πολλές διαστάσεις παραμένουν αδιαμέριστες, μεγαλώνοντας έτσι το μήκος του μονοπατιού αναζήτησης.

CAN & Multiple Realities

- Ένας τρόπος αύξησης της ευρωστίας / ανθεκτικότητας είναι να θεωρήσουμε Πολλαπλές Πραγματικότητες (Multiple Realities)
  - Δεν έχουμε 1 αλλά m διαφορετικά συστήματα συντεταγμένων
  - Κάθε κόμβος έχει μια ζώνη για κάθε σύστημα συντεταγμένων
  - Έτσι έχουμε m αντίγραφα ευρετηρίου
  - Μείωση του μήκους του μονοπατιού αναζήτησης (επιλέγεται το σύστημα συντεταγμένων βάσει του οποίου η αναζητούμενη ζώνη είναι εγγύτερα)



Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (CAN-style). Το σύστημα **pSearch**

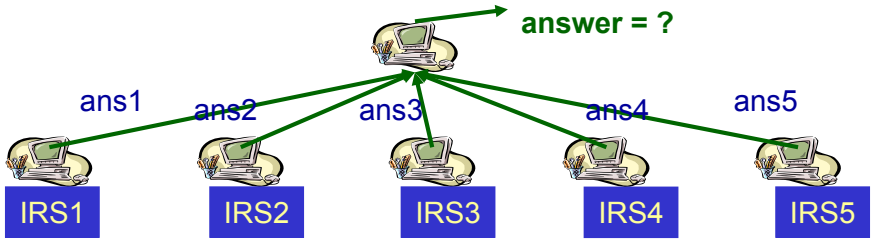
- Διαμερισμός των διανυσμάτων σε πολλά διανύσματα μικρότερης διάστασης
  - $(x_1, \ldots, x_n) \Rightarrow (x_1, \ldots, x_{n1}), (x_{n1+1}, \ldots, x_{n2}), (x_{n2+1}, \ldots, x_n)$
  - Τα πρώτα διανύσματα αποθηκεύονται σε ένα CAN1
  - Τα δεύτερα σε ένα CAN2, κ.ο.κ
  - Το διάνυσμα μιας επερώτησης επίσης διαμερίζεται σε διανύσματα μικρότερης διάστασης
  - :

Ανάκτηση Πληροφοριών σε **Δομημένα Ομ. Συσ/τα** (CAN-style). Το σύστημα **pSearch:** Σύνοψη

- Φτιάχνουμε ένα ευρετήριο όπου κάθε έγγραφο δεν περιγράφεται από το διάνυσμα του, αλλά από το διάνυσμα που προκύπτει αν πρώτα εφαρμόσουμε **Latent Semantic Indexing**
  - διανύσματα μικρότερης διάστασης, ομαδοποίηση εγγράφων
- Τα ευρετήριο αυτό διανέμεται στους κόμβους. Το κλειδί του κάθε εγγράφου είναι το διάνυσμα του (μετά την εφαρμογή του LSI). // Αυτό θα τοποθετήσει στον ίδιο κόμβο εννοιολογικά συναφή έγγραφα
- Ο υπολογισμός των διανυσμάτων απαιτεί καθολικά στατιστικά (άρα υπάρχει ανάγκη επικοινωνίας). Επίσης πρέπει να συμφωνηθεί η βάση των διανυσμάτων.

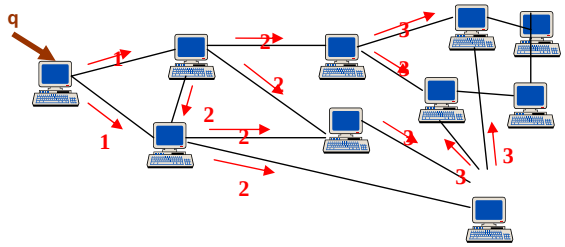
- Μπορεί να χρησιμοποιηθεί και για πολυμέσα (θυμηθείτε Feature-based Multimedia Indexing).

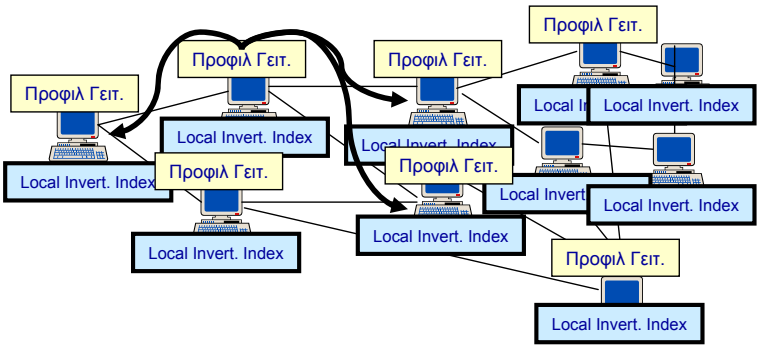
Ενοποίηση Αποτελεσμάτων & Ομότιμα Συστήματα



- Τεχνικές Ενοποίησης Αποτελεσμάτων
  - **Round Robin Inter-leaving**
  - **Score-based** (~ merge sort)
    - καλή αν τα σκορ υπολογίζονται βάσει των καθολικών στατιστικών
  - **Weighted-score based**
    - Έστω di προερχόμενο από μια πηγή Sj
    - score(di) = score(Sj,di) * score(Sj)
  - Λαμβάνοντας υπόψη μόνο τις διατάξεις και όχι τα σκορ (ενοποίηση διατάξεων)
    - **Borda, Condorcet, Kemeny**, Arrow's Impossibility Theorem

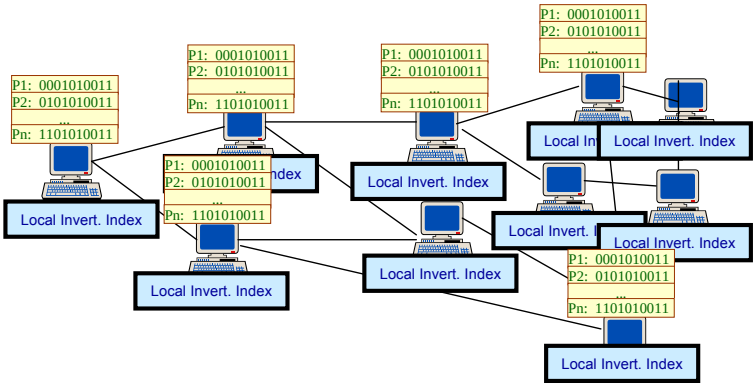
Ενοποίηση Αποτελεσμάτων σε Ομότιμα Συστήματα
(I): Οι κόμβοι δεν έχουν στη διάθεση τους **καθολικά** στατιστικά





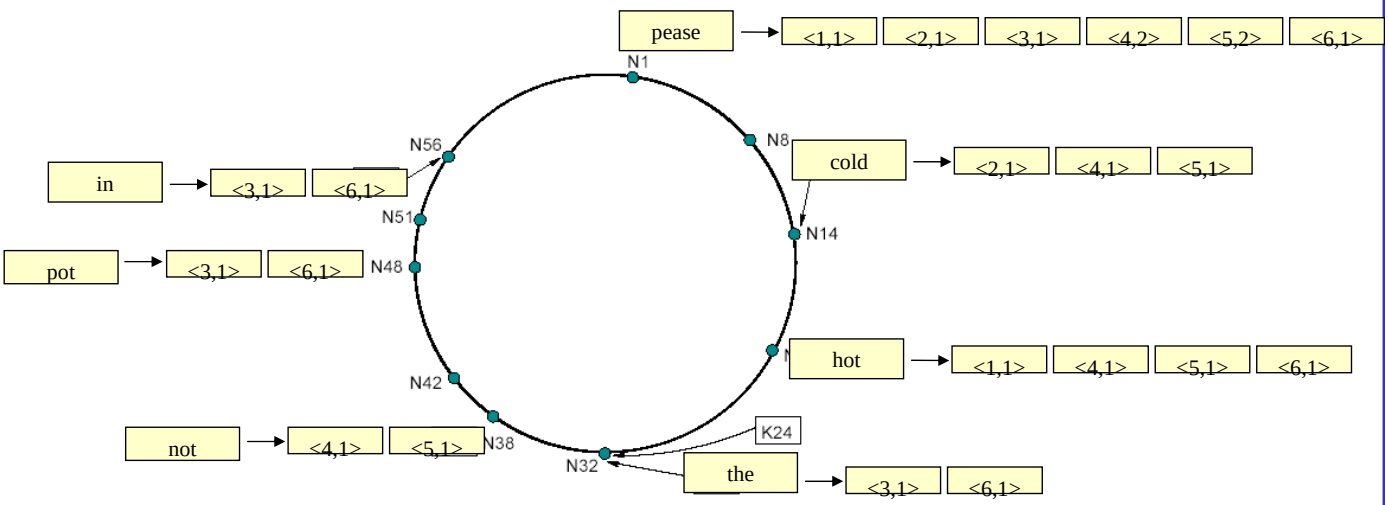
- Gnutella-like systems (document-partitioning):
  - Ενοποίηση: Round-robin interleaving, Score-based, Rank-Aggregation
- Συστήματα βασισμένα σε προφίλ γειτόνων και >RES
  - Ενοποίηση: Weighted score-based

Ενοποίηση Αποτελεσμάτων σε Ομότιμα Συστήματα
(II): Οι κόμβοι μπορούν να προσεγγίσουν τα **καθολικά** στατιστικά



- Π.χ. PlanetP (κάθε κόμβος μπορεί να προσεγγίσει το καθολικό ευρετήριο)
  - Ενοποίηση: Weighted score-based
    - (καλύτερο από το προφίλ γειτόνων, λιγότερα μηνύματα)

Ενοποίηση Αποτελεσμάτων σε Ομότιμα Συστήματα (III): Οι κόμβοι έχουν στη διάθεση τους τα **καθολικά** στατιστικά



- Π.χ. Chord-like (term-partitioning)
  - ο κόμβος που είναι υπεύθυνος για έναν όρο γνωρίζει τις συχνότητες εμφάνισης του καθώς και το πλήθος των κόμβων που έχουν έγγραφα που περιέχουν αυτόν τον όρο
  - Ενοποίηση: απλό Score-based είναι μια χαρά
    - κάθε κόμβος υπολογίζει partial scores, ο ερωτών τα αθροίζει και παράγει την τελική διάταξη

Ενοποίηση Αποτελεσμάτων σε Ομότιμα Συστήματα (III): Οι κόμβοι έχουν στη διάθεση τους τα **καθολικά** στατιστικά

- Έστω σύστημα όπως το Chord, στο οποίο τα κλειδιά είναι οι όροι και το οποίο συνολικά έχει 100.000 έγγραφα
- Η ανεστραμμένη λίστα ενός όρου έχει το πολύ 100.000 αναφορές σε έγγραφα (έστω ότι κατά μέσο όρο έχει 10.000 αναφορές)
- Έστω ότι ο p λαμβάνει επερώτηση q με 5 όρους. Κάθε όρος της q (μαζί με βάρος του στο q) θα προωθηθεί στον υπεύθυνο κόμβο για τον όρο αυτό
- Κάθε ένας από τους 5 κόμβους θα διατάξει τα έγγραφα βάσει του όρου αυτού και θα επιστρέψει μια λίστα μερικών αποτελεσμάτων
  - το πολύ 100.000 τριάδες (p, docId, score) κάτα μέσο όρο 10.000
- Ο p θα λάβει αυτές τις 5 λίστες και θα αθροίσει τα μερικά σκορ
  - score(doci) = score1(doci) + … + score5(doci)
- Άρα 5* 10.000 τριάδες ακεραίων πρέπει να μεταφερθούν στο δίκτυο
  - TotalBytes = 50K * 3 * 4 = 600 KB
- Ερώτηση: **Αν ο p θέλει να βρει μόνο τα κορυφαία κ (π.χ. κ=10) έγγραφα. Πως μπορούμε να ελαχιστοποιήσουμε την πληροφορία που πρέπει να μεταφέρουμε;**

Top-k Rank Aggregation

- Έχουμε **N** αντικείμενα και τους **βαθμούς** τους βάσει **m** διαφορετικών **κριτηρίων**.
- Έχουμε έναν τρόπο να συνδυάζουμε τα m σκορ κάθε αντικειμένου σε ένα ενοποιημένο σκορ
  - π.χ. min, avg, sum
- Στόχος: Βρες τα **κ** αντικείμενα με το υψηλότερο ενοποιημένο σκορ.

Εφαρμογές:

- Υπολογισμός των κορυφαίων-κ στοιχείων της απάντησης
  - ενός ΣΑΠ που βασίζεται στο διανυσματικό μοντέλο (τα m κριτήρια είναι οι m όροι της επερώτησης)
  - ενός μεσίτη πάνω από m Συστήματα Ανάκτησης Πληροφοριών
  - μιας επερώτησης σε μια Βάση Πολυμέσων
    - κριτήρια: χρώμα, μορφή, υφή, …

Άλλο ένα παράδειγμα εφαρμογής

- Ενοποίηση απαντήσεων σε Μεσολαβητές (middleware)
  - έστω μια υπηρεσία εύρεσης εστιατορίων βάσει τριών κριτηρίων:
    - τιμή γεύματος
    - απόσταση από ένα σημείο
    - κατάταξη εστιατορίου
  - όπου ο χρήστης μπορεί να ορίσει τον επιθυμητό τρόπο υπολογισμού του ενοποιημένου σκορ ενός εστιατορίου
    - π.χ. Σκορ= Τιμή*0.5 + Stars*0.25 + 0.25*DistanceFromHome
  - η υπηρεσία αυτή υλοποιείται με χρήση τριών απομακρυσμένων υπηρεσιών
    - (α) getRestaurantsByPrice
    - (b) getRestaurantsByStars
    - (c) getRestaurantsByDistance
  - **Πως μπορώ να ελαχιστοποιήσω το πλήθος των στοιχείων που πρέπει να διαβάσω από την απάντηση της κάθε υπηρεσίας, προκειμένου να βρω τα κορυφαία 5 εστιατόρια;**

Εύρεση των κ-κορυφαίων
Απλοϊκός Αλγόριθμος

1/ Ανέκτησε ολόκληρες τις m λίστες
2/ Υπολόγισε το ενοποιημένο σκορ του κάθε αντικειμένου
3/ Ταξινόμησε τα αντικείμενα βάσει του σκορ και επέλεξε τα πρώτα κ

Παρατηρήσεις
- Κόστος γραμμικό ως προς το μήκος των λιστών
- Δεν αξιοποιεί το γεγονός ότι οι λίστες είναι ταξινομημένες

Εύρεση των κ-κορυφαίων
Παράδειγμα: Απλοϊκός Τρόπος

S1 = < **A** 0.9, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >
S2 = < **B** 1.0, **E** 0.8, **F** 0.7, **A** 0.7, **C** 0.5, **H** 0.5, **G** 0.5 >
S3 = < **A** 0.8, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >

Ο Απλοϊκός Τρόπος

Score(**A**) = 0.9 + 0.7 + 0.8 = 2.4
Score(**B**) = 0.5 + 1.0 + 0.5 = 2
Score(**C**) = 0.8 + 0.5 + 0.8 = 2.1
Score(**E**) = 0.7 + 0.8 + 0.7 = 2.2
Score(**F**) = 0.5 + 0.7 + 0.5 = 1.7
Score(**G**) = 0.5 + 0.5 + 0.5 = 1.5
Score(**H**) = 0.5 + 0.5 + 0.5 = 1.5
Τελική διάταξη: < **A, E, C, B, F, G, H**>

Εύρεση των κ-κορυφαίων
Πιο Αποδοτικοί Αλγόριθμοι

- Γενική ιδέα: ***Άρχισε να διαβάζεις τις διατάξεις από την κορυφή. Προσπάθησε να καταλάβεις πότε πρέπει να σταματήσεις***.
- Αλγόριθμοι
  - **Fagin Algorithm (FA)** [Fagin 1999, J. CSS 58]
  - **Threshold Algorithm (TA)** [Fagin et al., PODS'2001]

Υποθέσεις

- Υποθέτουμε ότι έχουμε στη διάθεση μας 2 τρόπους πρόσβασης στα αποτελέσματα μιας πηγής:
  - **Σειριακή πρόσβαση** στις διατάξεις: φθίνουσα ως προς το σκορ
  - **Τυχαία προσπέλαση**: Δυνατότητα εύρεσης του σκορ ενός αντικειμένου με μία πρόσβαση
- Συναρτήσεις βαθμολόγησης (σκορ)
  - Τα σκορ ανήκουν στο διάστημα [0,1]
  - Η συνάρτηση ενοποιημένου σκορ είναι **μονότονη**
    - αν όλα (m) τα σκορ ενός αντικειμένου A είναι μεγαλύτερα ή ίσα των αντίστοιχων σκορ ενός αντικειμένου B, τότε σίγουρα το ενοποιημένο σκορ του A είναι μεγαλύτερο ή ίσο του σκορ του B

Εύρεση των κ-κορυφαίων
Ο Αλγόριθμος του Fagin (FA) [1999]

1.α/ Κάνε σειριακή ανάκτηση αντικειμένων από κάθε λίστα (αρχίζοντας από την κορυφή), έως ότου η τομή των αντικειμένων από κάθε λίστα να έχει κ αντικείμενα

1.β/ Για κάθε αντικείμενο που ανακτήθηκε (στο 1.α) συνέλεξε τα σκορ που λείπουν (με χρήση του μηχανισμού τυχαίας προσπέλασης)

2/ Υπολόγισε το ενοποιημένο σκορ του κάθε αντικειμένου

3/ Ταξινόμησε τα αντικείμενα βάσει του ενοποιημένου σκορ και επέλεξε τα πρώτα κ

Σχόλια

Αξιοποιεί (α) το γεγονός ότι οι λίστες είναι ταξινομημένες και (β) ότι η συνάρτηση ενοποίησης είναι μονότονη

[-] Το πλήθος των αντικειμένων που θα ανακτηθούν μπορεί να είναι μεγάλο

Εύρεση των κ-κορυφαίων
Παράδειγμα: Αλγόριθμος του Fagin (FA)

S1 = < **A** 0.9, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >
S2 = < **B** 1.0, **E** 0.8 **F** 0.7, **A** 0.7, **C** 0.5, **H** 0.5, **G** 0.5 >
S3 = < **A** 0.8, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >

Έστω ότι θέλω το Top-1

Το E εμφανίζεται σε όλες

(μονοτονία => δεν μπορεί κάποιο δεξιότερο του E να είναι καλύτερο του E

Το E δεν είναι σίγουρα ο νικητής.

Υποψήφιοι νικητές = {A, B, C, E, F}. Κάνουμε τυχαίες προσπελάσεις για να βρούμε τα σκορ που μας λείπουν

getScore(S2,A), getScore(S1,B), getScore(S3,B), getScore(S2,C), …

Πράγματι, top-1= {A}

Εύρεση των κ-κορυφαίων
Ο Αλγόριθμος TA (Threshold Algorithm) [Fagin et al. 2001]

Ιδέα:

Υπολόγισε το μέγιστο σκορ που μπορεί να έχει ένα αντικείμενο που δεν έχουμε συναντήσει ακόμα.

1/ Κάνε σειριακή ανάκτηση αντικειμένων από κάθε λίστα (αρχίζοντας από την κορυφή) και με χρήση τυχαίας προσπέλασης βρες όλα τα σκορ κάθε αντικειμένου

2/ Ταξινόμησε τα αντικείμενα (βάσει του ενοποιημένου σκορ) και κράτησε τα καλύτερα κ

3/ Σταμάτησε την σειριακή ανάκτηση όταν τα σκορ των παραπάνω κ αντικειμένων δεν μπορεί να είναι μικρότερα του μέγιστου πιθανού σκορ των απαρατήρητων αντικειμένων (threshold).

Εύρεση των κ-κορυφαίων
Παράδειγμα: Αλγόριθμος του Fagin (FA)

S1 = < **A** 0.9, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >
S2 = < **B** 1.0, **E** 0.8 **F** 0.7, **A** 0.7, **C** 0.5, **H** 0.5, **G** 0.5 >
S3 = < **A** 0.8, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >

Έστω ότι θέλω το Top-2

Το E, B (και το A) εμφανίζονται σε όλες

(μονοτονία => δεν μπορεί κάποιο δεξιότερο του B να είναι καλύτερο του B

Εύρεση των κ-κορυφαίων
Παράδειγμα: Αλγόριθμος TA:

S1 = < **A** 0.9, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >
S2 = < **B** 1.0, **E** 0.8 **F** 0.7, **A** 0.7, **C** 0.5, **H** 0.5, **G** 0.5 >
S3 = < **A** 0.8, **C** 0.8, **E** 0.7, **B** 0.5, **F** 0.5, **G** 0.5, **H** 0.5 >

Έστω ότι θέλω το Top-1

Score(A) = 0.9 + 0.7 + 0.8 = 2.4
Score(B) = 0.5 + 1.0 + 0.5 = 2
UpperBound = 0.9 + 1.0 + 0.8 = 2.7
αφού 2.7 > 2.4 συνεχίζω

Score(C) = 0.8 + 0.5 + 0.8 = 2.1
Score(E) = 0.7 + 0.8 + 0.7 = 2.2
UpperBound = 0.8 + 0.8 + 0.8 = 2.4
αφού 2.4 δεν είναι μεγαλύτερο του 2.4 (σκορ του A) σταματάω.

Σύγκριση: Fagin vs. TA

- Ο FA ποτέ δεν τερματίζει ενωρίτερα του TA
- Ο TA χρειάζεται μόνο έναν μικρό (k) ενταμιευτή (buffer)
- Ο TA μπορεί όμως να κάνει περισσότερες τυχαίες προσπελάσεις

Ο TA είναι βέλτιστος για όλες τις μονότονες συναρτήσεις σκορ

- Συγκεκριμένα, είναι "instant optimal": είναι καλύτερος πάντα (όχι μόνο στην χειρότερη περίπτωση ή στην μέση περίπτωση)

- Επεκτάσεις
  - Αλγόριθμος NRA (Non Random Access)
    - Έκδοση του TA για την περίπτωση που η τυχαία πρόσβαση είναι αδύνατη. Επίσης "instant optimal".
      - Do sequential access until there are *k* objects whose lower bound no less than the upper bound of all other objects
    - Αλγόριθμος CA (Combined Algorithm)
      - Έκδοση του TA που θεωρεί τις τυχαίες προσπελάσεις ακριβότερες των σειριακών.

Ενοποίηση Αποτελεσμάτων σε Ομότιμα Συστήματα (III): Οι κόμβοι έχουν στη διάθεση τους τα **καθολικά** στατιστικά

- Έστω σύστημα όπως το Chord, στο οποίο τα κλειδιά είναι οι όροι και το οποίο συνολικά έχει 100.000 έγγραφα
- Η ανεστραμμένη λίστα ενός όρου έχει το πολύ 100.000 αναφορές σε έγγραφα (έστω ότι κατά μέσο όρο έχει 10.000 αναφορές)
- Έστω ότι ο p λαμβάνει επερώτηση q με 5 όρους. Κάθε όρος της q (μαζί με βάρος του στο q) θα προωθηθεί στον υπεύθυνο κόμβο για τον όρο αυτό
- Κάθε ένας από τους 5 κόμβους θα διατάξει τα έγγραφα βάσει του όρου αυτού και θα επιστρέψει μια λίστα μερικών αποτελεσμάτων
  - το πολύ 100.000 τριάδες (p, docId, score) κάτα μέσο όρο 10.000
- Ο p θα λάβει αυτές τις 5 λίστες και θα αθροίσει τα μερικά σκορ
  - score(doci) = score1(doc1) + … + score5(doc1)
- Άρα 5* 10.000 τριάδες ακεραίων πρέπει να μεταφερθούν στο δίκτυο
  - TotalBytes = 50K * 3 * 4 = 600 KB
- Ερώτηση: **Αν ο p θέλει να βρει μόνο τα κορυφαία κ (π.χ. κ=10) έγγραφα. Πως μπορούμε να ελαχιστοποιήσουμε την πληροφορία που πρέπει να μεταφέρουμε;**

- Διαφορές με Κατανεμημένη Ανάκτηση
- Napster-style
- Gnutella-style (local inv. Index)
- Freenet-style
  - (p,q,|ans(q)|), >RES
  - (p,q,|ans(q)|), >RES * sim(q)
- Hiearchical
- PlanetP (Bloom filters)
- Chold-style: key=1 term, a term pair, …
  - term partitioning
- CAN-style: key = LSI vector pSearch (LSI + CAN)
  - document partitioning
  - Result aggregation

Γενικά: P2P & IR = αντικείμενο έρευνας σήμερα

Αναφορές

- D. Zeinalipour-Yazti, Vana Kalogeraki, Dimitrios Gunopulos, Information Retrieval in P2P Networks
- Text-Based Content Search and Retrieval in *ad hoc* P2P Communities, Francisco Matias Cuenca-Acuna and Thu D. Nguyen
- Jie Lu, Jamie Callan, «Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks», SIGMOD'04 workshop
- Fagin, Lotem, and Naor, Optimal Aggregation Algorithms for Middleware (PODS 2001)