HY463 - Συστήματα Ανάκτησης Πληροφοριών
Information Retrieval (IR) Systems

Μέρος Γ

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

Γιάννης Τζίτζικας

Διάλεξη : 17b
Ημερομηνία : 30-5-2007

Συστήματα Ομοτίμων
(Peer-to-Peer Systems)

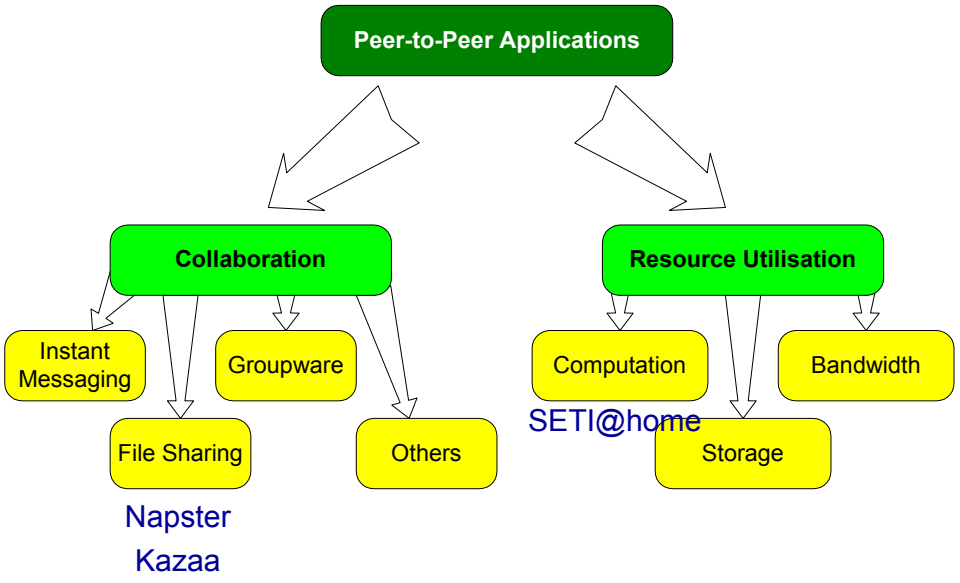
Διάρθρωση

- 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)

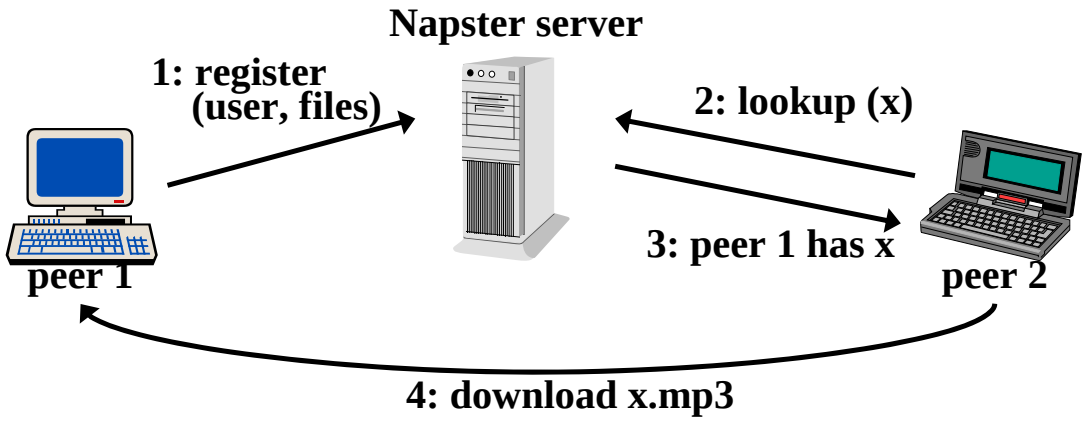
P2P: Εφαρμογές



Ομότιμα Συστήματα 1ης Γενιάς: Υβριδικά
Napster

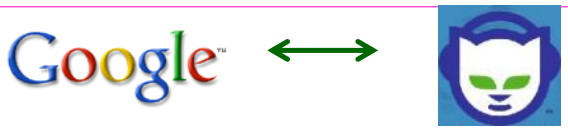
Ονομάζονται και **Υβριδικά Ομότιμα Συστήματα** (Hybrid P2P systems) διότι υπάρχει ένας κεντρικός εξυπηρετητής

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

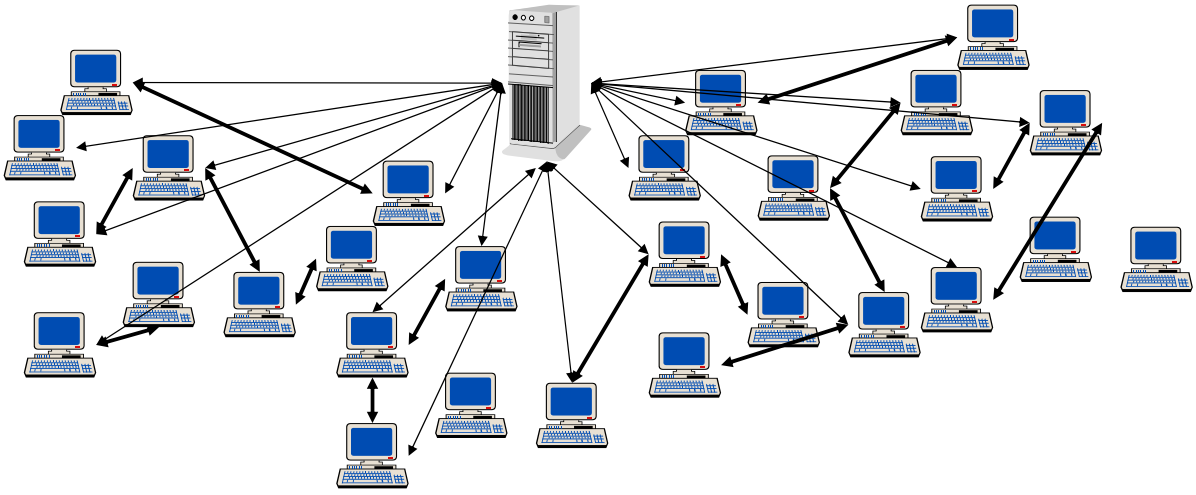
Διαμερισμός πόρων

- **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

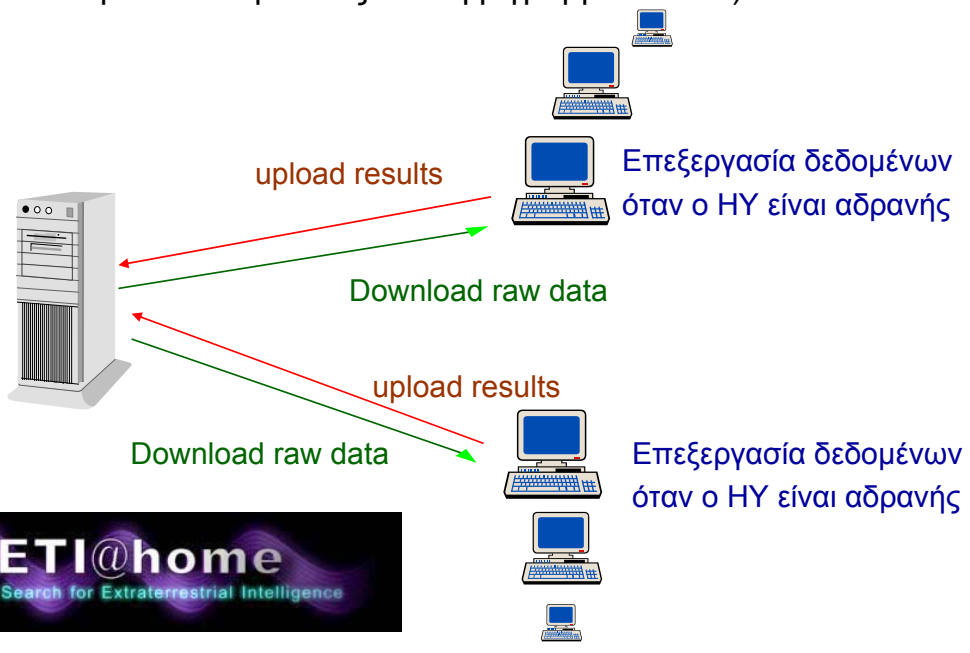
Ομότιμα Συστήματα 1ης Γενιάς: Υβριδικά
Napster



A central point of failure

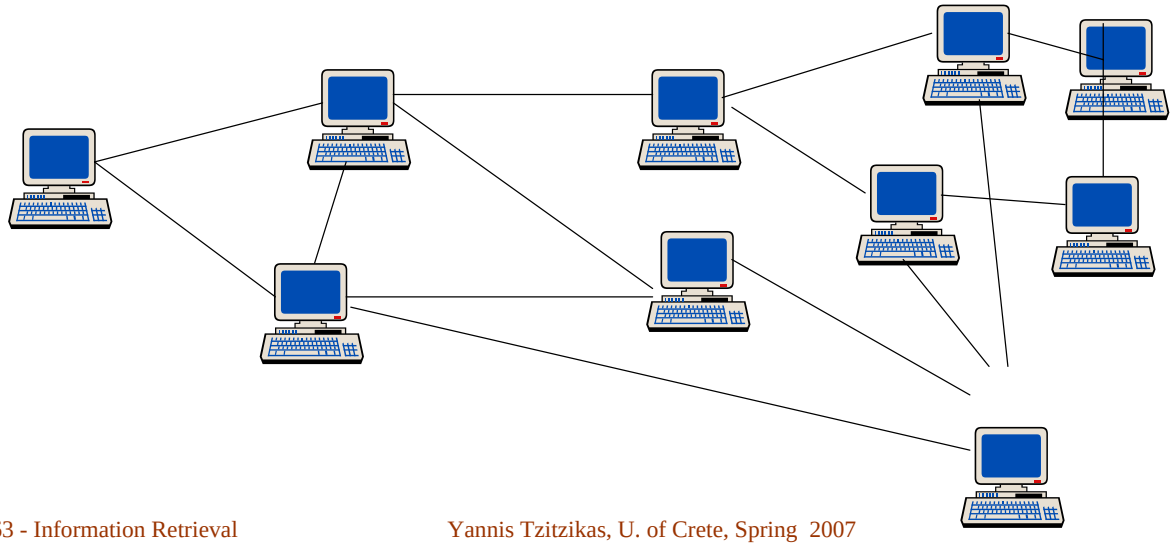
Ομότιμα Συστήματα 1ης Γενιάς: Υβριδικά
SETI@home

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



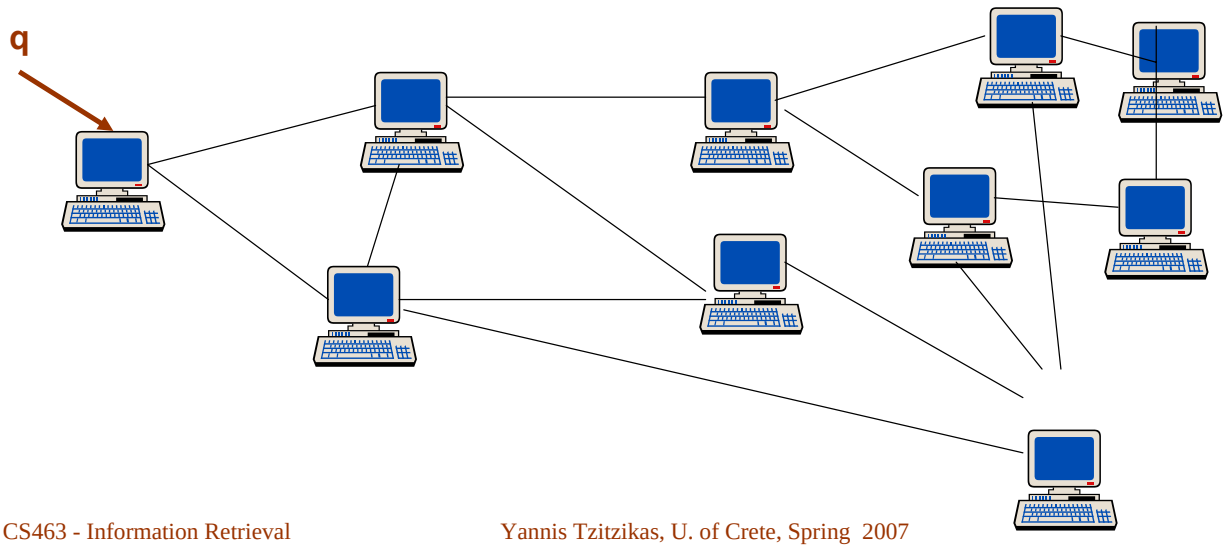
Ομότιμα Συστήματα 1ης Γενιάς: **Αποκεντρωμένα GNUTELLA**

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

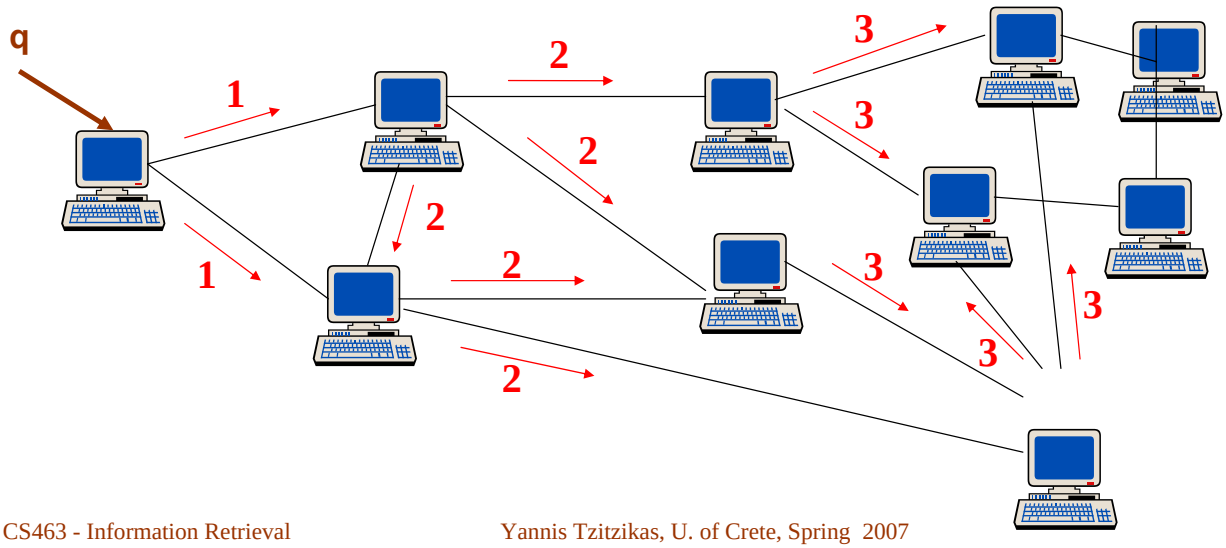


Ομότιμα Συστήματα 1ης Γενιάς: **Αποκεντρωμένα GNUTELLA**

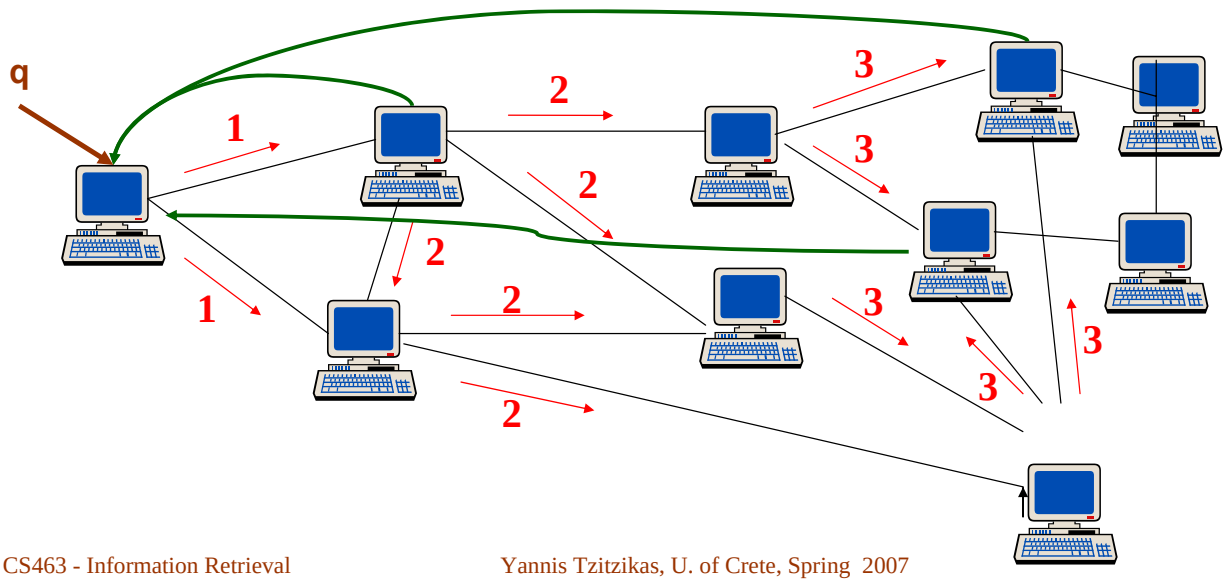
- Δεν υπάρχει **κανένας κεντρικός εξυπηρετητής**
- Ονομάζονται και Αποκεντρωμένα (Decentralized P2P systems), Αδόμητα (Unstructured P2P systems), Pure P2P systems
- Gnutella (1999-now):



Κατακλυσμός Μηνυμάτων (Message Flooding or Gossiping)

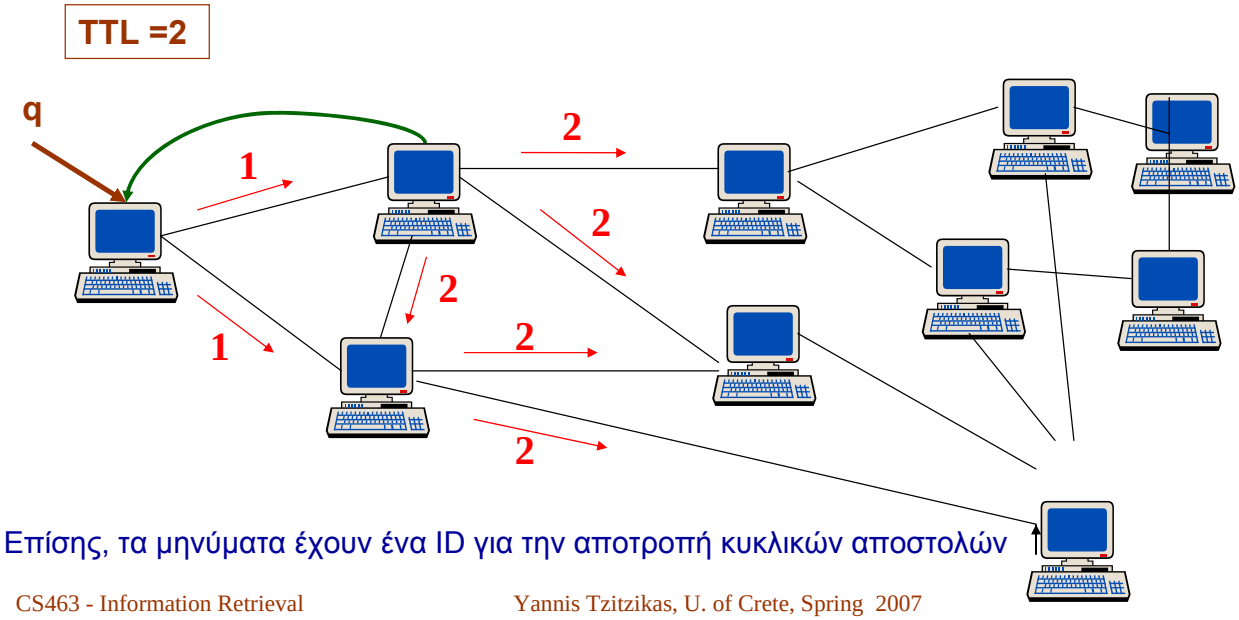


Κατακλυσμός Μηνυμάτων (*Message Flooding*)



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

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

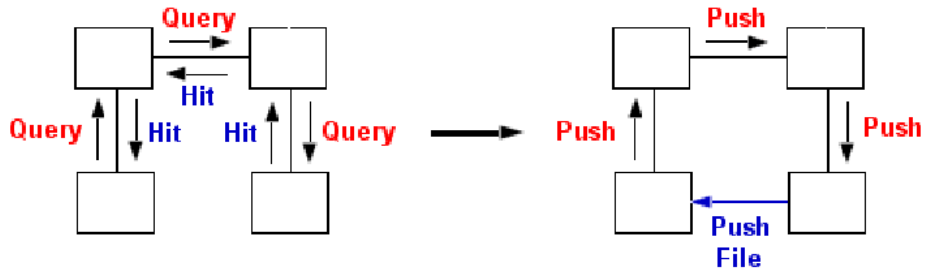


Επίσης, τα μηνύματα έχουν ένα ID για την αποτροπή κυκλικών αποστολών

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:

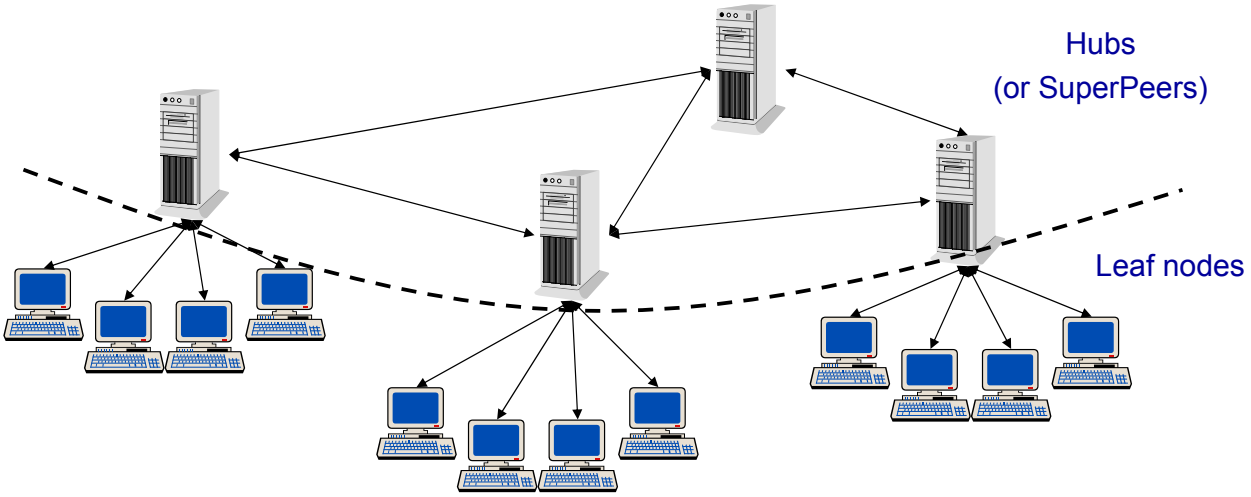
Chord (MIT), CAN (Berkeley), OceanStore/Tapestry (Berkeley), Farsite (MSR), Spinglass/Pepper (Cornell), Pastry/PAST (Rice, MSR), Viceroy (Hebrew U), P-Grid (EPFL), P2P-Net (Magdeburg), Pier (Berkeley), Peers (Stanford), Kademlia (NYU), Bestpeer (Singapore), YouServ (IBM Almaden), Hyperion (Toronto),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

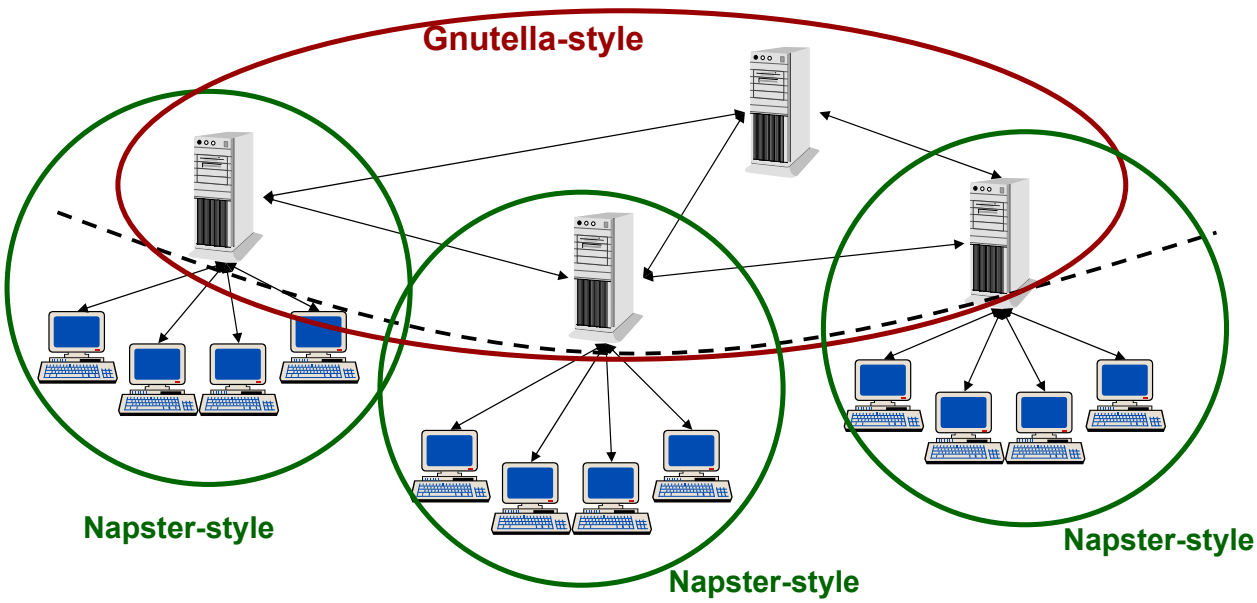
Ιεραρχικά Ομότιμα Συστήματα (**Hierarchical** P2P Systems)

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



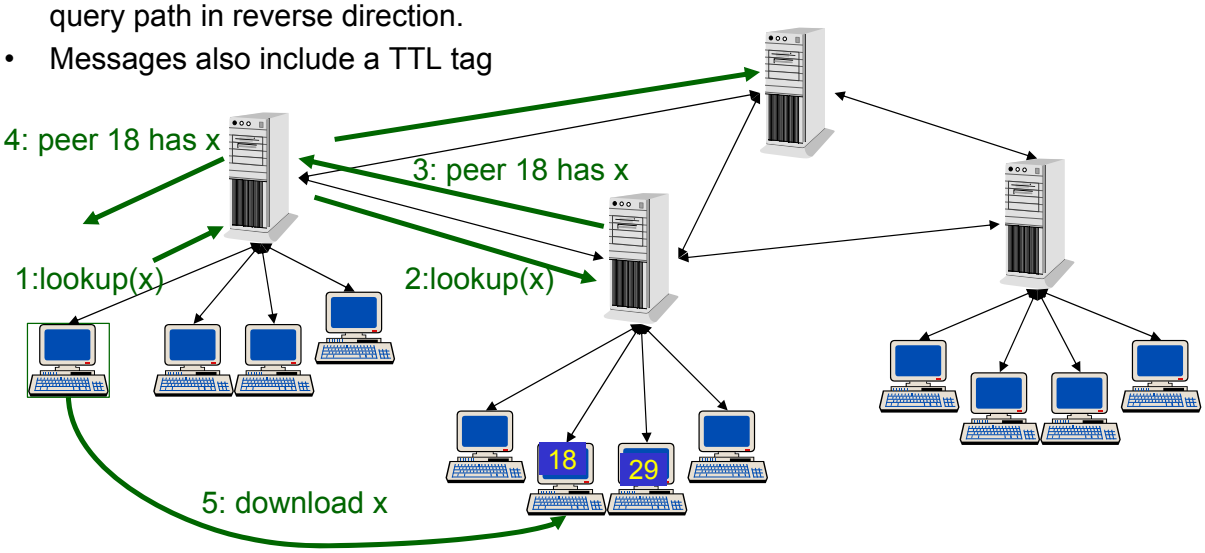
Ιεραρχικά Ομότιμα Συστήματα (**Hierarchical** P2P Systems)

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**

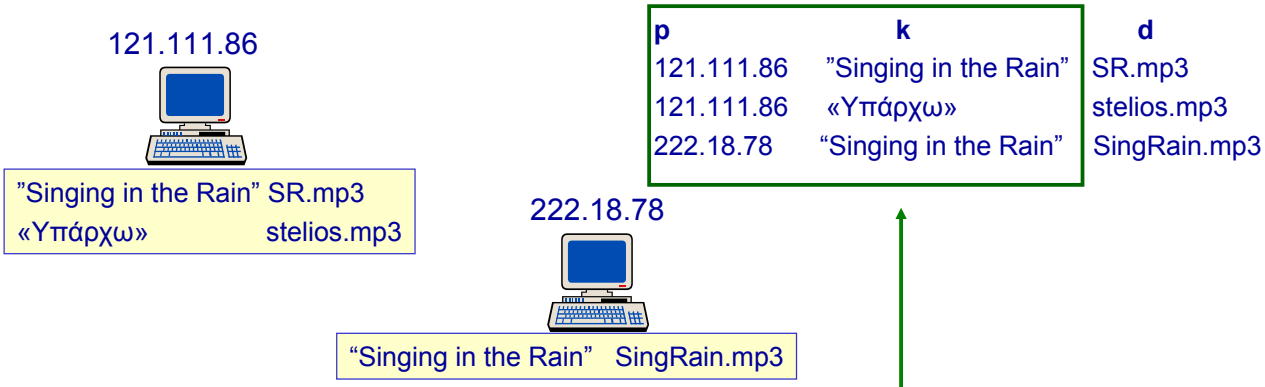
- Συστήματα
  - Freenet, Chord, CAN, Pastry, Tapestry, FreeNet, P-Grid,

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).**

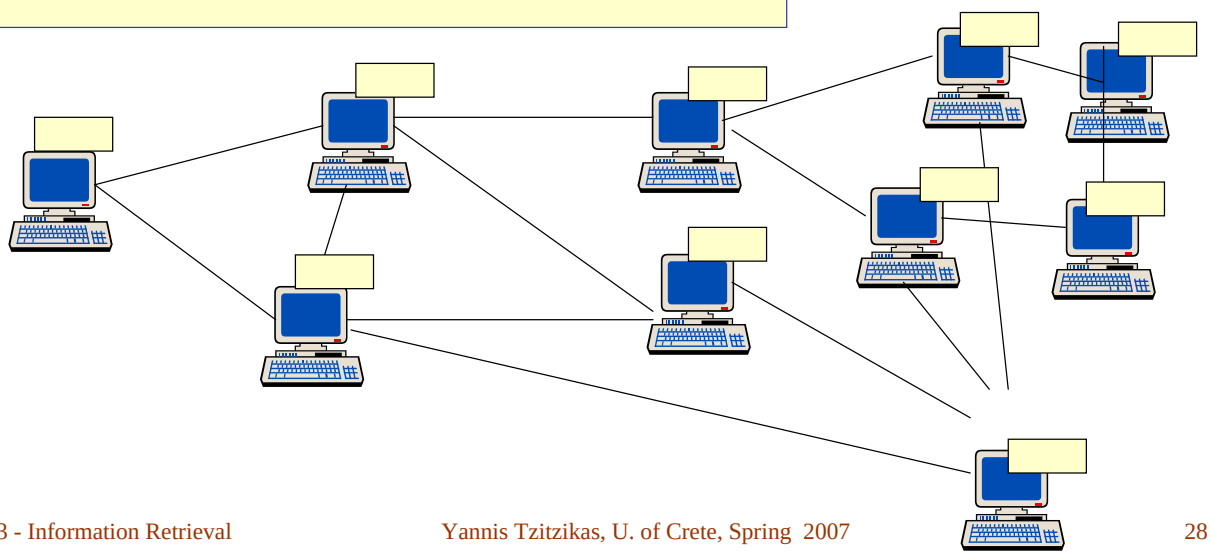
(άρα το ευρετήριο μας αποτελείται από ζεύγη της μορφής **(k,p))**

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

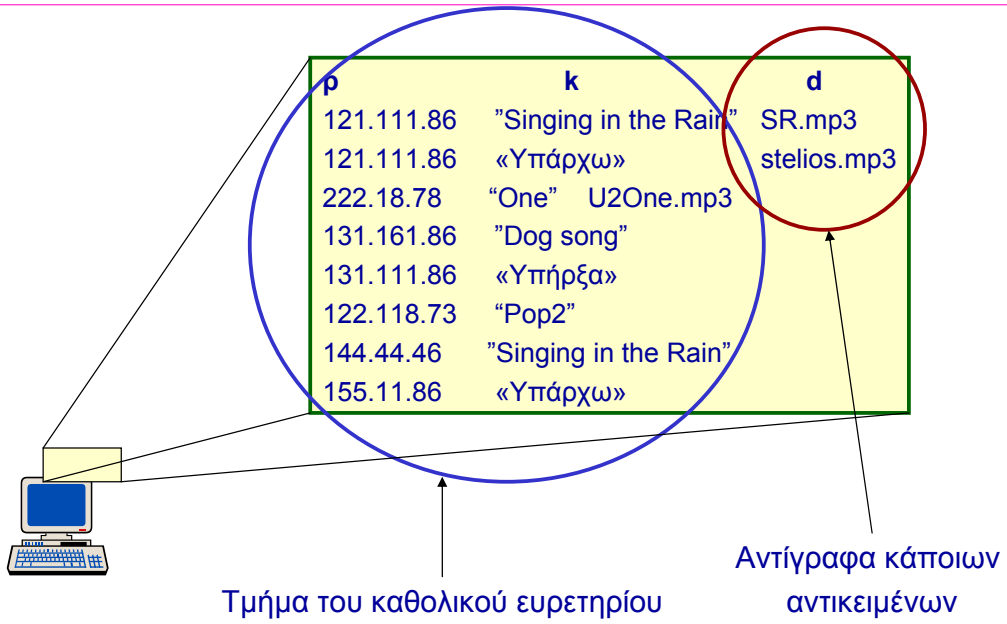
Freenet:

- 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



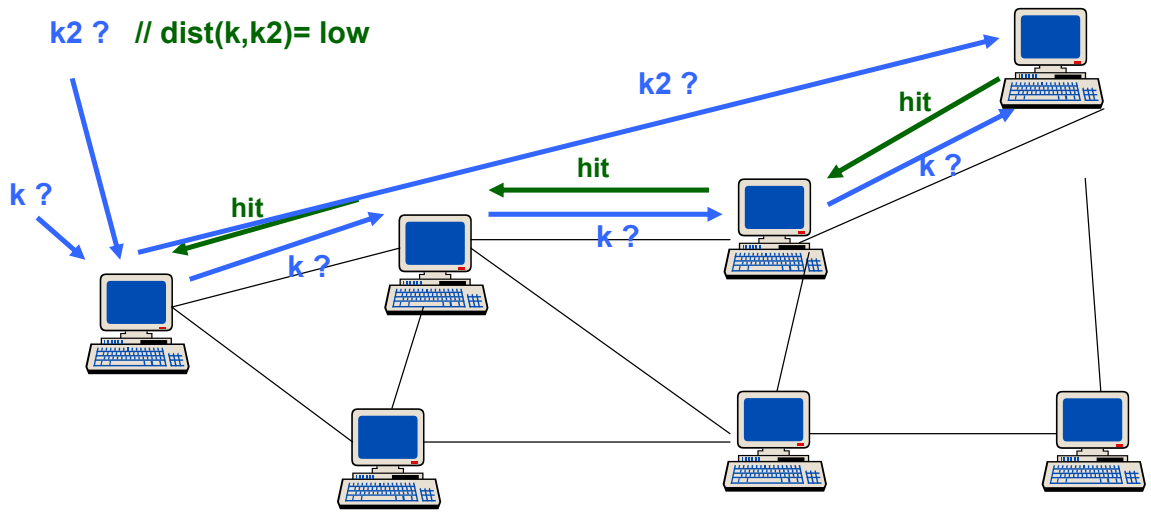
Freenet: Cache



Freenet: Τρόπος Εντοπισμού Πόρου

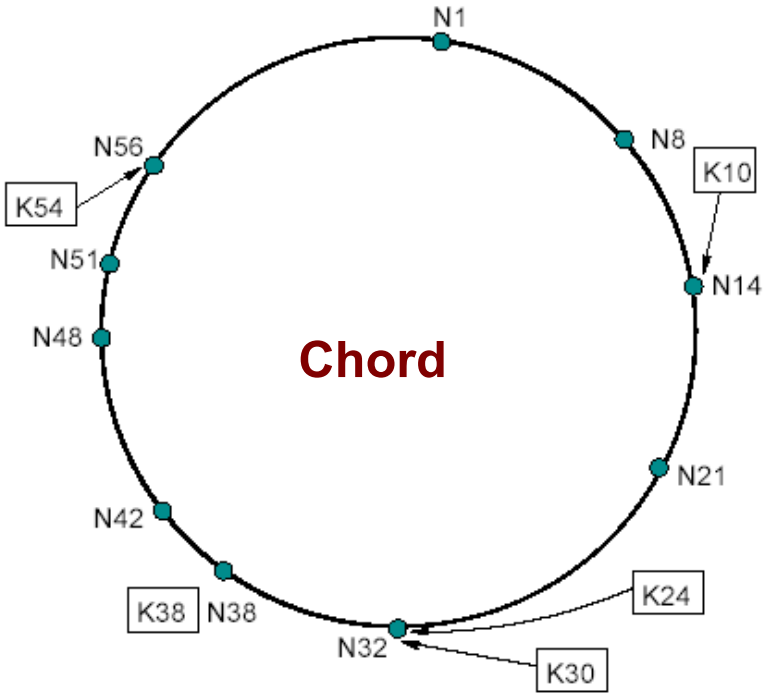
- Έλευση επερώτησης k
  - Αν η έγγραφή (p,k,d) είναι στη κρυφή μνήμη επέστρεψε το d
  - Αλλιώς προώθησε την επερώτηση στον κόμβο που έχει το **πιο όμοιο κλειδί**
- Η διαδικασία αυτή συνεχίζεται με αυτόν τον τρόπο έως ότου ευρεθεί το αναζητούμενο ή το TTL φτάσει την τιμή 0.
- Έλευση απάντησης (k,p,d)
  - 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



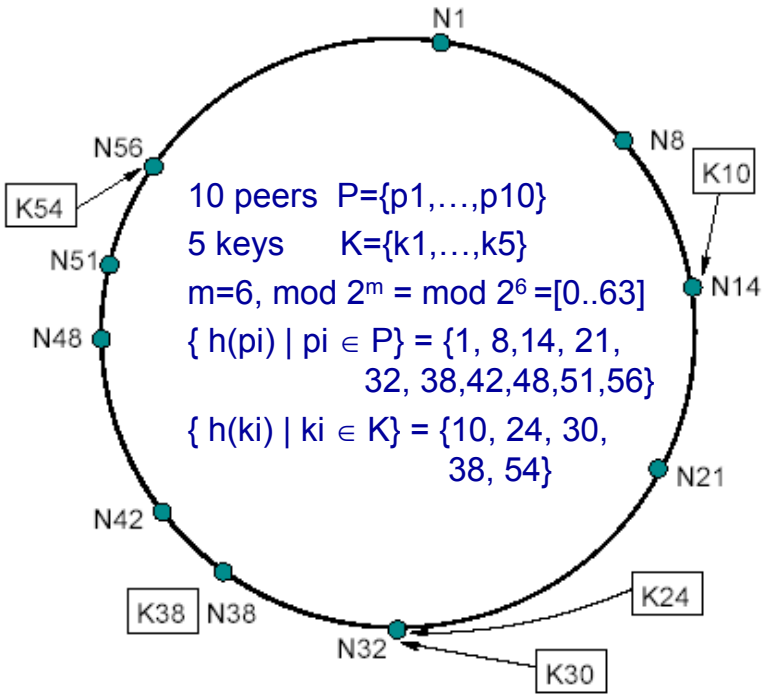
Freenet: Εισαγωγή Νέου Πόρου

- 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))

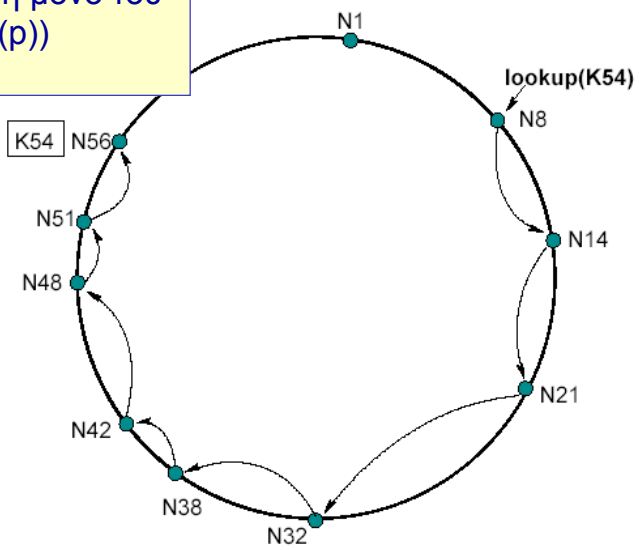
- Κατακερματισμός (Hashing) κλειδιών (k)και διευθύνσεων (p) σε δυαδικά κλειδιά με **m**-bits
  - π.χ. m=6, **h**(«υπάρχω»)=**11**, **h**(196.178.0.1)=**3**
- Τα δυαδ. κλειδιά τοποθετούνται σε έναν κύκλο modulo 2^m
  - Για m=8, κυκλική διάταξη των αριθμών 0…255
- Ένα κλειδί **k** εκχωρείται στον πρώτο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

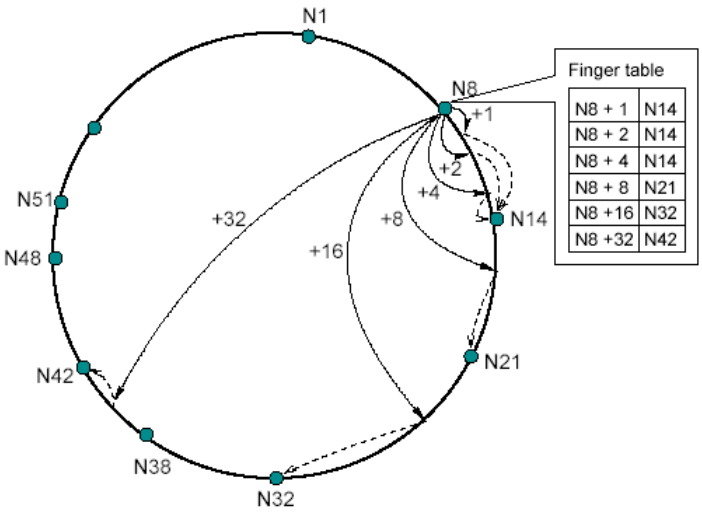
- Επιπλέον πληροφορία δρομολόγησης για επιτάχυνση
- Κάθε κόμβος n έχει έναν **πίνακα δρομολόγησης** με m εγγραφές
  - οι m αυτοί κόμβοι έχουν εκθετικά αυξανόμενη απόσταση από τον n
- Η i εγγραφή του πίνακα έχει την δνση του πρώτου κόμβου με κλειδί μεγαλύτερο ή ίσο με $n+2^{i-1}$

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

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

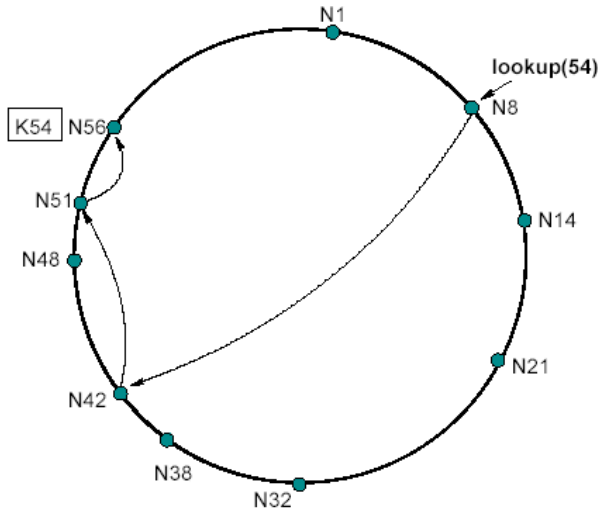
βρίσκει τον μικρότερο peer με κλειδί μεγαλύτερο αυτού της επερώτησης.

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

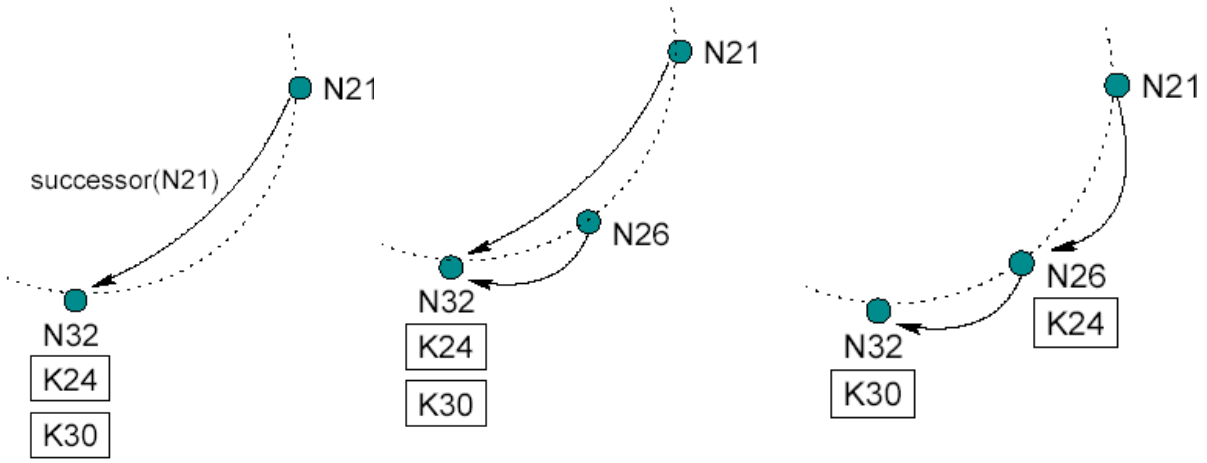
N42+ 1	N48
N42+ 2	N48
N42+ 4	N48
N42+ 8	N51
N42+16	N1
N42+32	N14

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

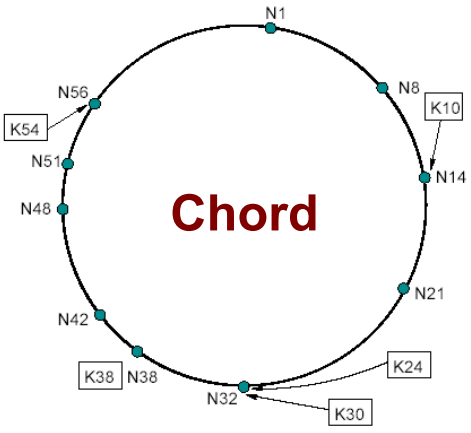
Βασικά σημεία:

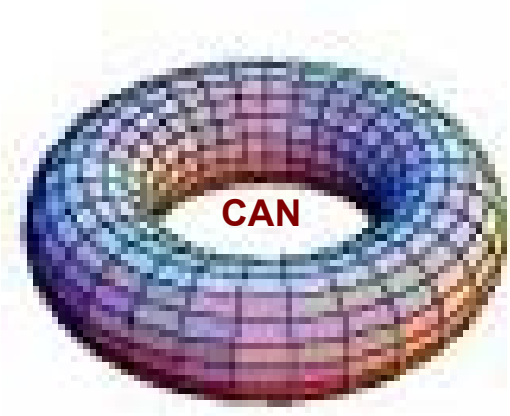
- Κάθε κόμβος αποθηκεύει πληροφορία για μικρό αριθμό κόμβων (m)
  - Κάθε κόμβος ξέρει περισσότερα για τους κοντινούς του όρους (απ'ότι για τους μακρινούς)

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)

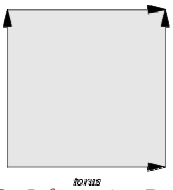
- Chord software
  - 3000 lines of C++ code, Library to be linked with the application, provides a lookup(key) – function: yields the IP address of the node responsible for the key, Notifies the node of changes in the set of keys the node is responsible for

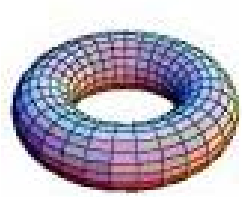




Δομημένα Ομότιμα Συστήματα
CAN (Content Addressable Network)

- 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

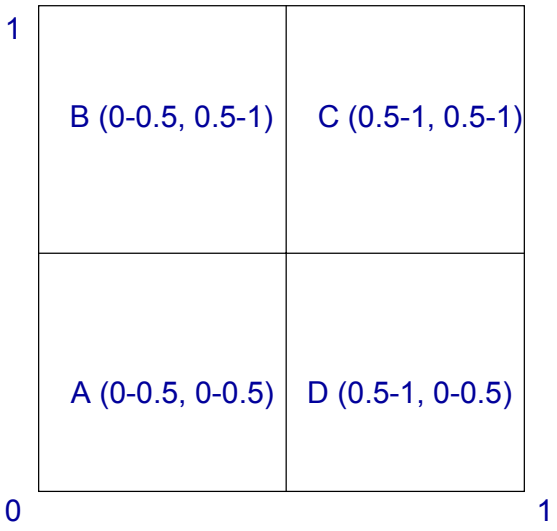




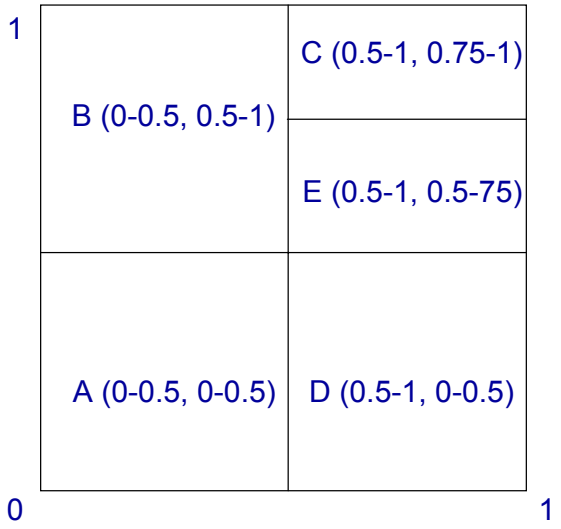
http://mathworld.wolfram.com/Torus.html

Δομημένα Ομότιμα Συστήματα
CAN

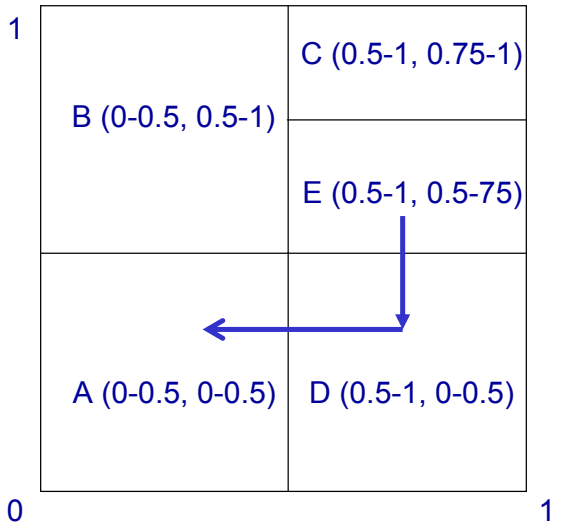
- Π.χ. για 2D, 4 peers A, B, C, D

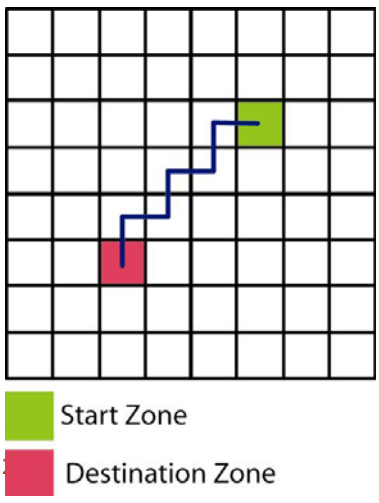


- Είσοδος ενός νέου κόμβου E



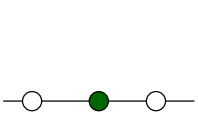
- Ο E θέλει να ανακτήσει το αντικείμενο με συντεταγμένες (0.2, 0.2)

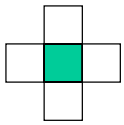


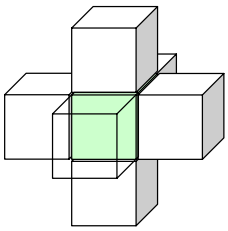


Δομημένα Ομότιμα Συστήματα
CAN

- 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

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

Δομημένα Ομότιμα Συστήματα
CAN Περίληψη

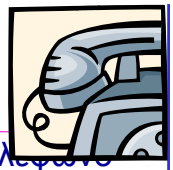
Βασικά σημεία:

- 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

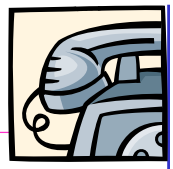


Έστω ότι στην Ελλάδα κατοικούν 1000 άτομα και κάθε ένας τους έχει ένα τηλέφωνο

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