9 IR in Peer-to-Peer Systems

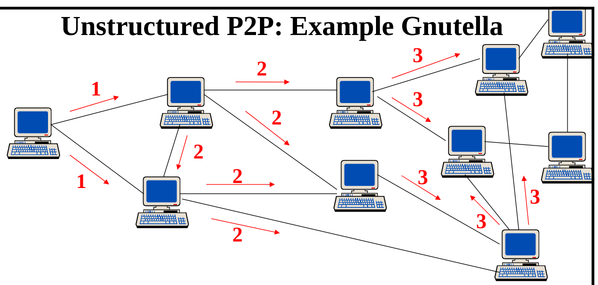
9.1 Peer-to-Peer (P2P) Architectures

Decentralized, self-organizing, highly dynamic loose coupling of many autonomous computers

Applications:
- **Large-scale distributed computation (SETI, PrimeNumbers, etc.)**
- **File sharing (Napster, Gnutella, KaZaA, etc.)**
- **Publish-Subscribe Information Sharing (Marketplaces, etc.)**
- **Collaborative Work (Games, etc.)**
- **Collaborative Data Mining**
- **(Collaborative) Web Search**

Goals:
- **make systems ultra-scalable and completely self-organizing**
- **make complex systems manageable and less susceptible to attacks**
- **break information monopolies, exploit small-world phenomenon**

Unstructured P2P: Example Gnutella



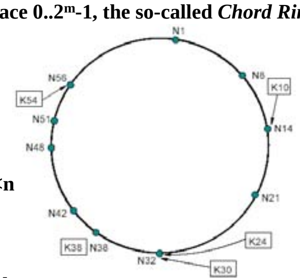
all forward messages carry a TTL tag (time-to-live)

1) **contact neighborhood and establish virtual topology (on-demand + periodically): *Ping, Pong***
2) **search file: *Query, QueryHit***
3) **download file: *Get* or *Push* (behind firewall)**

Structured P2P: Example Chord

Distributed Hash Table (DHT):
map strings (file names, keywords) and numbers (IP addresses) onto very large „cyclic" key space 0..2^m-1, the so-called *Chord Ring*

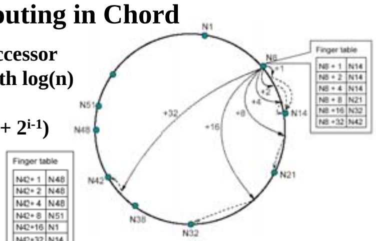
Key k (e.g., hash(file name)) is assigned to the node with key n (e.g., hash(IP address)) such that k ≤ n and there is no node n' with k ≤ n' and n'<n



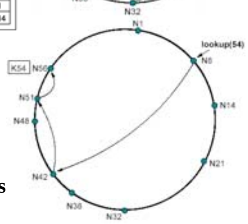
Properties & claims:
Unlimited scalability (> 10^6 nodes)
O(log n) hops to target, O(log n) state per node
Self-stabilization (many failures, high dynamics)

Request Routing in Chord

Every node knows its successor and has a finger table with log(n) pointers: finger[i] = successor (node number + 2^{i-1}) for i=1..m



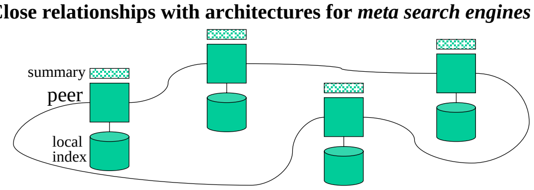
For finding key k perform recursively: determine current node's largest finger[i] (modulo 2^m) with finger[i] ≤ k



Successor ring and finger tables require dynamic maintenance

9.2 Query Routing

Close relationships with architectures for *meta search engines* !



If I want to submit a query to k<<n peers, where should I send it?

<u>Architectural approach:</u>
- **every peer posts (statistical) summary info about its contents**
- **query routing is driven by query-summaries similarities**
- **summaries are organized into a distributed registry**
  - **maintained at selected super-peers**
  - **embedded into DHT**
  - **lazily replicated at all peers (via „gossiping")**

Differences between Meta and P2P Search Engines

Meta Search Engine	P2P Search Engine
small # sites (e.g., digital libraries)	huge # sites
rich statistics about site contents	poor/limited/stale summaries
static federation of servers	highly dynamic system
each query fully executed at each site	single query may need content from multiple peers
interconnection topology largely irrelevant	highly dependent on overlay network structure

Random Query Routing (RAPIER)

Peer selection for given query driven by (query-independent) *„possession rules“*, e.g., each peer has partial information about a conceptually global term-peer matrix $D_{m \times n}$ with D_{ij} = 1 iff peer j has non-empty index list for term i

<u>RAPIER (Random Possesion Rule):</u>

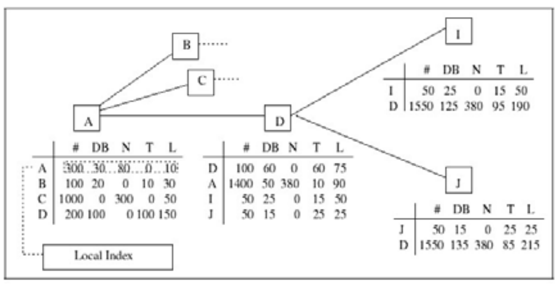
- **peers forward queries along unstructured P2P network**
- **choose random item i with non-zero entry in local D**
- **randomly choose k peers with non-zero entries of i^{th} row of local D, possibly biased with probabilities ~ $\|D_{*j}\|_1$**

Alternative:
view each row of local D as a „shopping basket“
perform association rule mining to determine „guide rules“
of the form: peer x, peer y, peer z ⇒ peer w

Routing Indices

Every peer (in an acyclic overlay-network topology*) maintains summary information about each of its neighbors:

- **the total number of docs held by the neighbor and all nodes transitively reachable from the neighbor together**
- **the same for particular topics or topic sets**



from: Arturo Crespo, Hector Garcia-Molina: Routing Indices for Peer-to-Peer Systems, ICDCS 2002

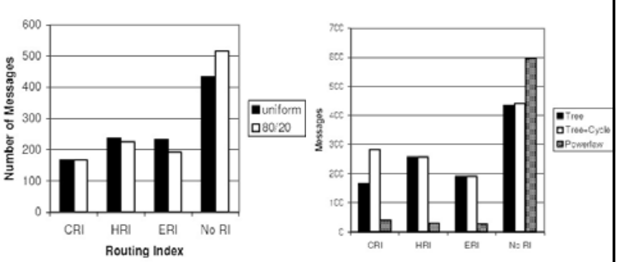
*** unclear how exactly non-tree topologies are handled**

Simulation of Routing Indices (1)

Compound RIs: total #docs in reachable peers (goodness)
Hop-count RIs: goodness of distance-i reachable peers (i=1,2, ...)
Exponential RIs: $\sum_i \sum_{n \in \text{distance-i peers}}$ {goodness(n)/fanout^i}

Parameter Name	Description	Value
Network Configuration Parameters		
$NumNodes$	Number of nodes in the network	60000
T	Topology of the P2P network	tree
F	Branching factor for tree topology	4
EL	Extra links for tree+cycle topology	10
o	Outdegree exponent for power law	-2.2088
Document Distribution Parameters		
QR	Total Number of Query Results	3125
D	Document distribution	80/20
RI Parameters		
$Creation_{size}$	Avg. size of creation/update message	1000 b
$Query_{size}$	Avg. size of query message	250 b
$StopCondition$	Number of documents requested	10
H	Horizon for HRIs	5
A	Decay for ERs	4
c	RI Compression	0%
$minUpdate$	min % diff for update propagation	1%

Simulation of Routing Indices (2)



from: Arturo Crespo, Hector Garcia-Molina: Routing Indices for Peer-to-Peer Systems, ICDCS 2002

Query Routing based on IPF (PlanetP)

Every peer conceptually maintains the inverse peer frequency (IPF) for each term i:

$$IPF_i = \log\left(1 + \frac{\#\,peers}{\#\,peers\ with\ term\ i}\right)$$

For multi-keyword query q the quality of peer j is:

$$R_j(q) := \sum_{i \in q} IPF_i \cdot \begin{cases} 1 & if\ \ peer\ \ j\ contains\ term\ i \\ 0 & otherwise \end{cases}$$

To retrieve top k results for query q:

1. **rank peers in descending order of Rj(q)**
2. **contact peers in groups of m in rank order**
3. **merge results**
4. **iterate steps 2 and 3 until no peer contributes to top-k result**

PlanetP Implementation

Each peer posts its summary in the form of a *Bloom-filter signature*:

- **bit vector S[1..s] of fixed length s, initially all bits zero**
- **if peer j has term i it sets bit h(i) to one using a hash function h**
- **other peers can test if peer j holds term set {q1, ..., qk} by looking up S[h(q1)], ..., S[h(qk)] or by computing a bit vector Q[1..s] for {q1, ..., qk} and ANDing S with Q, both with the risk of *„false positives“***

Summaries are sent to other peers by asynchronous *gossiping* in a combined push/pull mode:

- ***push*: periodically send updates of global registry (small Δs) as „rumors“ to randomly chosen neighbors; stop doing so when n consecutive peers already know the update**
- **(anti-entropy) *pull*: periodically ask randomly chosen neighbor to send an updated summary of the global registry; alternatively ask push-sender for recent rumors**

Query Routing based on Similarity Measures

For query q select peers p with highest value of sim(q, p), e.g., cosine(q, p) where p is represented by its centroid

Use statistical language model for similarity:

$$KL(q \| p) = \sum_{t \in q} P[t \mid q] \log \frac{P[t \mid q]}{\lambda P[t \mid C_p] + (1-\lambda) P[t \mid G]}$$

where P[t|q], P[t|Cp], P[t|G] are the (estimated) probabilities that term t is generated by the language models for the query q, the corpus C_p of peer p, and the general vocabulary, and λ is a smoothing parameter between 0 and 1

The Kullback-Leibler divergence (aka. relative entropy) is a measure for the distance between two probability distributions:

$$KL(f \| g) := \sum_x f(x) \log \frac{f(x)}{g(x)}$$

Query Routing based on Goodness (GlOSS)

Goodness (q, s, l) = $\sum$ {sim(q, d) | d $\in$ result(q, s) $\wedge$ lsim(q,d)>l} for query q, source s, and score threshold l

GlOSS (Glossary Of Servers Server) aims to rank sources by goodness

Approximate goodness by using for source s:

- **$df_i(s)$: number of docs in s that contain term i**
- **$w_i(s)$: $\sum$ {$tf_i(d)$*idf_i | d $\in$ s} (total weight of term i in s)**

High-correlation assumption:
$df_i(s) \le df_j(s) \Rightarrow$ every doc in s that contains i also contains j

Uniformity assumption:
$w_i(s)$ is distributed uniformly over all docs in s that contain i

Goodness with High-correlation Assumption

For fixed source s and query q = t_1 ... t_n with $df_i \le df_{i+1}$ for i=1..n-1 consider subqueries q_p = t_p ... t_n (p=1..n).
Every doc d in s that contains t_p ... t_n has query similarity

$$sim_p(q,d) = \sum_{j=p..n} t_j \frac{w_i(s)}{df_i(s)}$$

Find smallest p such that $sim_p(q,d)$>l and $sim_{p+1}(q,d) \le$ l

EstGoodness(q,s,l) = $\sum_{j=1..p}$ ($df_j(s)$ – $df_{j-1}(s)$) * sim_j

Goodness with Disjointness Assumption

Disjointness assumption:
{d$\in$s|d contains term i} $\cap$ {d$\in$s|d contains term j} = $\emptyset$ for all i,j $\in$q

Uniformity assumption:
$w_i(s)$ is distributed uniformly over all docs in s that contain i

$$sim(q,d) = \sum_{j=1..n} t_j \frac{w_i(s)}{df_i(s)}$$

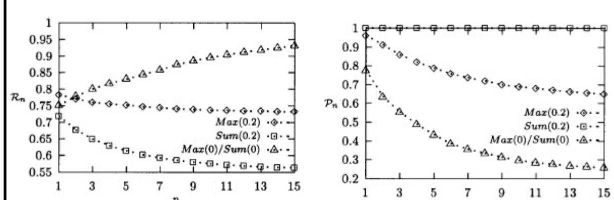
EstGoodness(q,s,l) = $\sum_{j=1..n \wedge sim>l} df_i(s) \cdot t_j \frac{w_i(s)}{df_i(s)}$

$$= \sum_{j=1..n \wedge sim>l} t_j \; w_i(s)$$

GlOSS Experiments (1995)

evaluation metrics for top-n source ranking:
R_n := $\sum_{i=1..n}$ {estGoodness(i[th] rank) / Goodness(i[th] rank)}
P_n := |{s|estGoodness(s) in top-n $\wedge$ Goodness(s)>l}| / n

6800 newsgroup user profiles as queries over
53 different newsgroups (comp.databases, comp.graphics, rec.arts.cinema, ...)



from: L. Gravano, H. Garcia-Molina, A. Tomasic: GlOSS: Text-Source Discovery over the Internet, ACM TODS 24(2), 1999

Usefulness Estimation Based on MaxSim

Def.: A set S of sources is ***optimally ranked*** **for query q in the order s1, s2, ..., sm if for every n>0 there exists k, 0<k≤m, such that s1, ..., sk contain the n best matches to q and each of s1, ..., sk contains at least one of these n matches**

Thm.: **Let MaxSim(q,s) = max{sim(q,d)|q∈s}. s1, ..., sm are optimally ranked for query q if and only if MaxSim(q,s1) > MaxSim(q,s2) > ... > MaxSim(q,sm).**

Practical approach („Fast-Similarity method"):
Capture, for each s, dfi(s), $avgw_i(s)$, $maxw_i(s)$ as source summary.
Estimate for query q = t1 ... tk

$$MaxSim(q,s) := \max_{i=1..k} \{t_i * maxw_i(s) + \sum_{v \neq i} t_v * avgw_v(s)\}$$

estimation time linear in query size,
space for statistical summaries linear in #sources * #terms

9.3 Distributed Query Execution Issues

Algorithm:
- **Determine the number of results to be retrieved from each source a priori based on the source's content quality vs.**
- **Run distributed version of Fagin's TA**

Dynamic adaptation:
- **Plan query execution only once before initiating it vs.**
- **Dynamic plan adjustment based on sources' result quality and responsiveness (incl. failures)**

Parallelism:
- **Start querying all selected sources in parallel vs.**
- **Consider (initial) results from one source when querying the next sources**

9.4 Result Reconciliation

Case 1: all peers use the same scoring function, e.g. cosine similarities based on tf*idf weights

Case 2: peers may use different scoring functions that are publicly known

Case 3: peers may use different & unknown scoring functions but provide scored results

Case 4: peers provide only result rankings, no scores

Techniques for Result Reconciliation (1)

for case 1:

local sim is

$$lsim(\vec{q},\vec{d}) = \sum_i \frac{q_i \cdot tf_i(\vec{d}) \cdot lidf_i}{\sqrt{\sum_i q_i^2} \cdot \sqrt{\sum_i tf_i(\vec{d})^2 \cdot lidf_i^2}}$$

global sim is

$$sim(\vec{q},\vec{d}) = \sum_i \frac{q_i \cdot tf_i(\vec{d}) \cdot gidf_i}{\sqrt{\sum_i q_i^2} \cdot \sqrt{\sum_i tf_i(\vec{d})^2 \cdot gidf_i^2}}$$

submit additional single-term queries (one for each query term) such that each result d to the original query q is retrieved:

$$lsim(q_i,\vec{d}) = \frac{q_i \cdot tf_i(\vec{d}) \cdot lidf_i}{q_i \cdot \sqrt{\sum_j tf_j(\vec{d})^2 \cdot lidf_j^2}} = \frac{tf_i(\vec{d}) \cdot lidf_i}{\sqrt{\sum_j tf_j(\vec{d})^2 \cdot lidf_j^2}}$$

$$\Rightarrow \frac{lidf_i}{\sqrt{\sum_j tf_j(\vec{d})^2 \cdot lidf_j^2}} = \frac{lsim(q_i,\vec{d})}{tf_i(\vec{d})}$$

Techniques for Result Reconciliation (2)

for case 4:

set global score of doc j retrieved from source i to

$$g(d_j) := 1 - (r_{local}(d_j) - 1) \cdot \frac{r_{\min}}{m \cdot r_i}$$ **where**

- **$r_{local}(dj)$ is the local rank of d_j,**
- **r_i is the score of source i among the queried sources,**
- **r_{min} is the lowest such score, and**
- **m is the number of desired global results**

Intuition:
- **initially local ranks are linearly mapped to scores**
- **the factor r_{min} / (m r_i) is the score difference for consecutive ranks from source i**

Literature (1)

- Communications of the ACM, Vol 46, No. 2, Special Section on Peer-to-Peer Computing, February 2003.
- Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan: Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications, To Appear in IEEE/ACM Transactions on Networking.
- F.M. Cuenca-Acuna, C. Peery, R.P. Martin, T.D. Nguyen: PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, IEEE Symp. on High Performance Distributed Computing, 2003
- Jie Lu, Jamie Callan: Content-Based Retrieval in Hybrid Peer-to-Peer Networks, CIKM Conference, 2003.
- Edith Cohen, Amos Fiat, Haim Kaplan; Associative Search in Peer to Peer Networks: Harnessing Latent Semantics, INFOCOM, 2003
- Mayank Bawa, Roberto J. Bayardo Jr., Sridhar Rajagopalan, Eugene Shekita: Make it Fresh, Make it Quick - Searching a Networks of Personal Webservers, WWW Conference, 2003.

Literature (2)

- Arturo Crespo, Hector Garcia-Molina: Routing Indices for Peer-to-Peer Systems, ICDCS Conf. 2002
- Luis Gravano, Hector Garcia-Molina, Anthony Tomasic: GlOSS: Text-Source Discovery over the Internet, ACM TODS Vol.24 No.2, 1999
- Weiyi Meng, Clement Yu, King-Lup Liu: Building Efficient and Effective Metasearch Engines, ACM Computing Surveys Vol.34 No.1, 2002
- Clement Yu, King-Lup Liu, Weiyi Meng, Zonghuan Wu, Naphtali Rishe: A Methodology to Retrieve Text Documents from Multiple Databases, IEEE TKDE Vol.14 No.6, 2002
- Norbert Fuhr: A Decision-Theoretic Approach to Database Selection in Networked IR, ACM TOIS Vol.27 No.3, 1999
- Henrik Nottelmann, Norbert Fuhr: Evaluating Different Methods of Estimating Retrieval Quality for Resource Selection, SIGIR 2003