Peer-to-Peer Replication Strategies

Ransom Briggs

Peer-to-Peer Definition

- Client – Server scalability problems
- Spread load over many computers
- Each peer has equivalent capabilities
- Adaptable network protocols

P2P Evolution

- Centralized P2P Systems
  - Napster
- Decentralized P2P Systems
  - Unstructured
    - Gnutella, Freenet
  - Structured
    - Pastry, Tapestry, Skipnet, CAN, Chord

Problem Formation

- How to place replicas
  - Reduce search latency
  - Reduce load on hotspots
- Side affects not considered
  - Fault tolerance
  - File availability

Peer-to-Peer Replication

- Unstructured P2P Background
  – C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks."
- Unstructured P2P Replication Strategies
  – Y. Chawathe, S. Ratnasamy, L. Breslau, and S. Shenker. "Making Gnutella-like P2P Systems Scalable."
  – Cohen, E. and Shenker, S. "Replication Strategies in Unstructured Peer-to-Peer Networks."
  – Kamal Jain, Vijay V. Vazirani. "Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems."

Peer-to-Peer Replication (cont'd)

- Structured P2P Background
  - –A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for largescale peer-to-peer systems."
- Structured P2P Replication Strategies
  - –S. Iyer, A. Rowstron, P. Druschel. "Squirrel: A decentralized, peer-to-peer Web cache."
  - –Y. Chen, R. H. Katz, and J. D. Kubiatowicz. "Dynamic replica placement for scalable content delivery."
  - –Venugopalan Ramasubramanian and Emin Gun Sirer. "Beehive: O(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays."

Unstructured P2P Systems

- Network setup of loosely associated peers
- Each peer knows of only a few peers
- File searches executed by searching peers
- Each peer evaluates a query against index

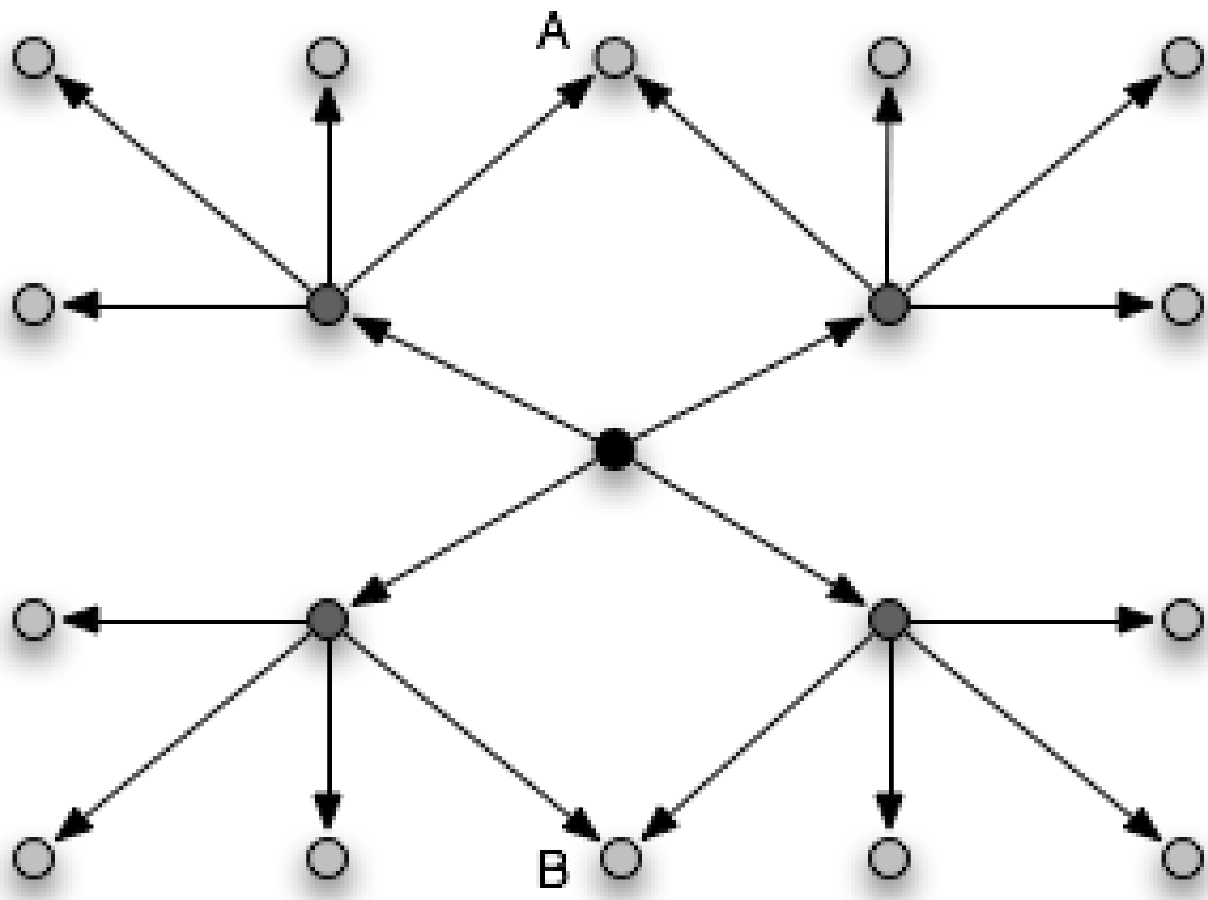
Unstructured P2P Systems

- Returns successful to searching peer
- Each query has a time-to-live counter
- Systems differ in how a query is forwarded

Query Flooding

- Recursively forwards query to neighbors
- Sends unnecessary duplicate messages
- Number of peers visited per round increases exponentially w/ respect to degree

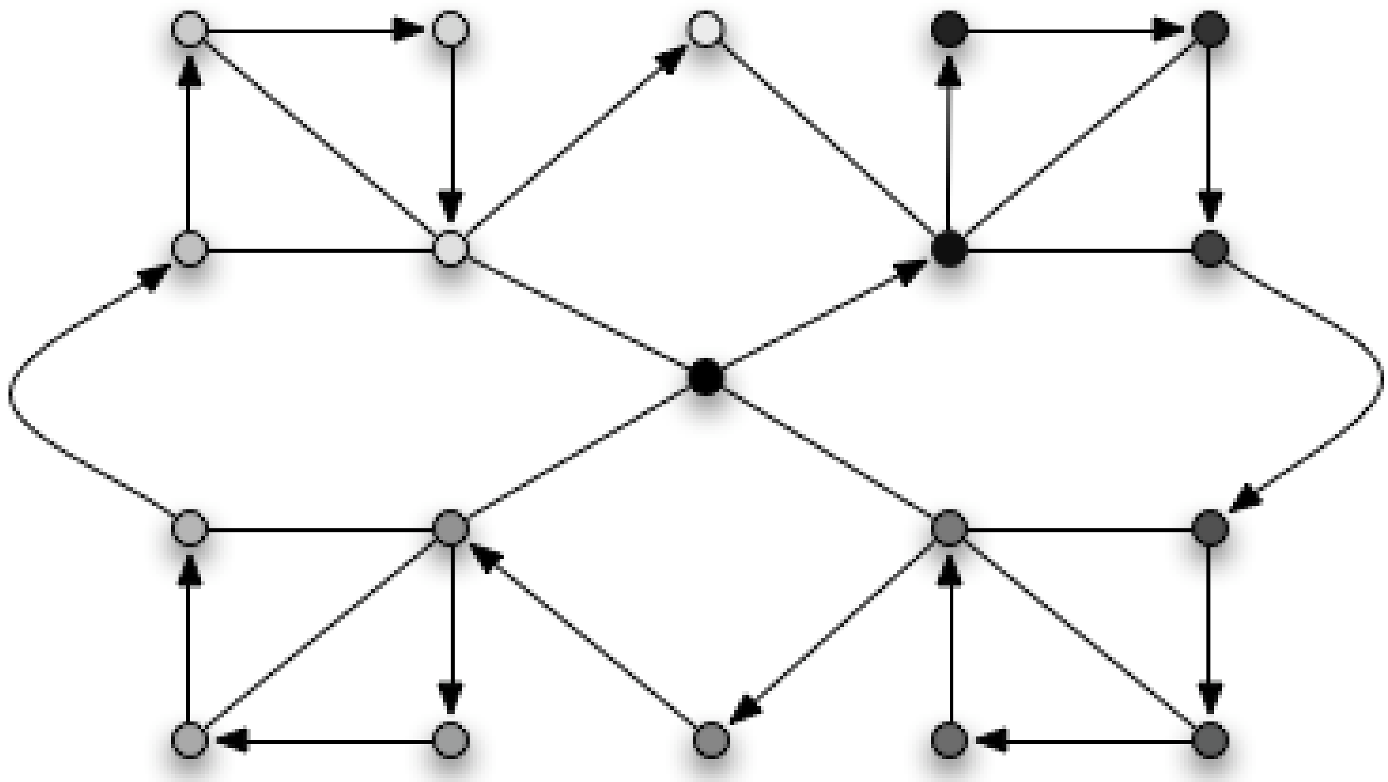
Flooding Illustration



Random Walker

- Sends out a query that is randomly forwarded to a single neighbor
- Multiple walkers can be sent out so that a greater number of peers are visited
- Less likely to duplicate messages

Random Walker Illustration



Peer-to-Peer Replication

- Unstructured P2P Background
- Unstructured P2P Replication Strategies
- Structured P2P Background
- Structured P2P Replication Strategies

Unstructured System Gia

- Gia
  - – Exploit peer heterogeneity
  - – Number of links varies by peer capacity
  - – Capacity is generic, a mix of network, disk and processor capacity
  - – Queries are routed towards higher capacity neighbors

Gia Replication

- Each peer indexes of all its files
- The indices of peers are replicated with the owner's IP address
- Indices are replicated at all neighbors
- Higher capacity peers
  – have more neighbors
  – larger aggregated indices
  – more queries routed

Unstructured P2P Model

- "Replication Strategies in Unstructured Peer-to-Peer Networks." by Edith Cohen and Scott Shenker
- n = number of peers in the network
- m = number of distinct data items
- r_i = number of replicas for file i
- Assumed that each peer has an equal probability that it contains a replica
- Probability that any particular peer has file i will be r_i / n

Unstructured P2P Model (cont'd)

- Assumes a geometric distribution
- Expected number peers visited to find a file (A_i) equals n / r_i
- q_i - fraction of all queries for file *i*
  - $\sum q_i = 1$
- Average search size over all file requests is weighted by the query rate
- $AverageSearchSize = \sum q_i * (n / r_i)$

Replication Allocations

- ρ = number of replicas each peer can store
- R = total number of replicas in the system

  $R = n * \rho$
- Objective: minimize *AverageSearchSize* by adjusting r_i

Uniform & Proportional Allocations

- AverageSearchSize = $\sum q_i * (n / r_i)$
- $R = n\rho, \sum q_i = 1$
- Uniform Allocation
  - Replicate all objects uniformly
  - Sets $r_i = R / m$
  - *AverageSearchSize* = m / ρ
- Proportional Allocation
  - Replicate all objects relative to q_i
  - *Sets* $r_i = R * q_i$
  - *AverageSearchSize* = m / ρ

Square Root Allocation

- Minimized the *AverageSearchSize*
- $r_i = (R / \sum sqrt(q_i))\ sqrt(q_i)$
- $AverageSearch = (1 / \rho)\ (\sum sqrt(q_i))^2$

Replication Model / Path Replication

- Upon successful search the client creates C copies of found file
- Let $\langle C_i \rangle$ be average C used for file i
- Path Replication
  - $r_i / (n\rho) \propto q_i \langle C_i \rangle$
  - $A_i \propto 1 / (q_i \langle C_i \rangle \rho)$
  - Fixed point when $A_i = \langle C_i \rangle \propto 1/sqrt(q_i)$
- Set C to be the search size

Sibling Neighbor Memory

- Path Replication overshoots square root
- Adjust *C* value to account for previous object creation
- FIFO cache replacement policy
  - Replica existence probability decreases with time
  - LRU will not work

Replication with Probe Memory

- Receive query for file *i*
  - Record search size for the query
  - Attach the search size to the query
  - Aggregate across multiple nodes
  - Better estimate actual q_i and r_i

Utilization Rate

- $U_i = q_i / (r_i / R)$
- Average utilization rate is same for all
- Maximum varies among allocations
  - Uniform: proportional to query rate
  - Proportional: perfect utilization
  - Square Root: Falls in between two strategies

Replica Placement

- Owner Replication
  - Implicit replication
- Path Replication
  - Replicas placed along successful search
- Random Replication
  - Replicas placed randomly among searched peers

k-median Problem

- Bi-partite graph - facilities and customers
- Edges between facilities and customers is the cost of connecting
- Open *k* facilities minimizing the total cost of connecting all customers
- Analogous to placing *k* replicas minimizing the network cost

k-median Problems

- Exact solution NP-hard
- Centralized solution
  - 6 approximate
  - *O(edge log edge (log (peers))*
- Decentralized solutions
  - Non constant approximations
  - Network overhead is prohibitive

Peer-to-Peer Replication

- Unstructured P2P Background
- Unstructured P2P Replication Strategies
- Structured P2P Background
- Structured P2P Replication Strategies

Structured P2P Systems

- Given message and key, routes to node responsible for the key
- Each peer assigned an ID
- Routes in a guaranteed number of logical hops

Pastry Details

- Hyper Cube routing
  - Routes in $\log_b N$ logical hops
- Routing table with *log(b) N* rows and *b* columns
  - *b* is the base of the identifier
  - Row i denotes that the peer shares i prefixes and differs at i+1
  - Column j denotes that peer has digit j at i+1
- Stores *L* closest IDs in leaf set

Pastry Routing Table and Leaf Set

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

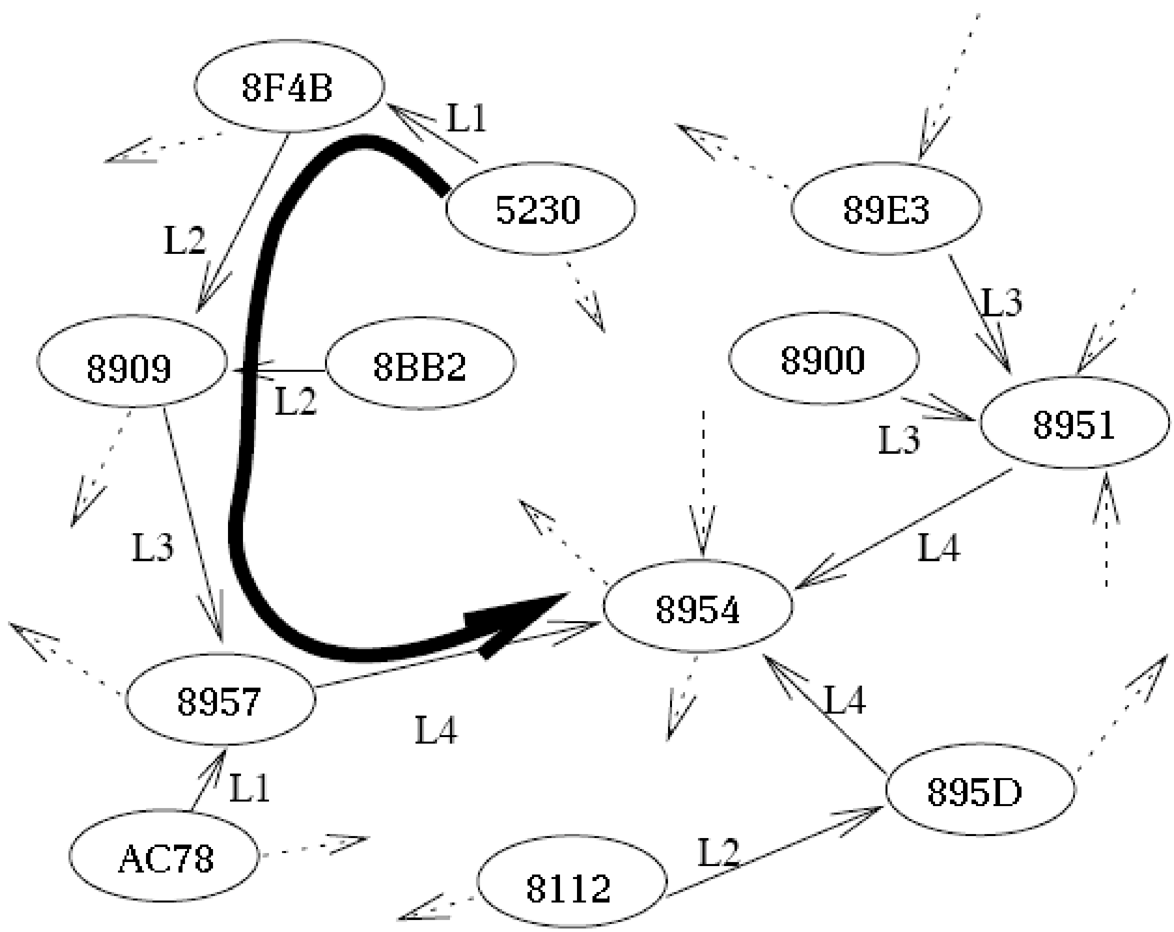
Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

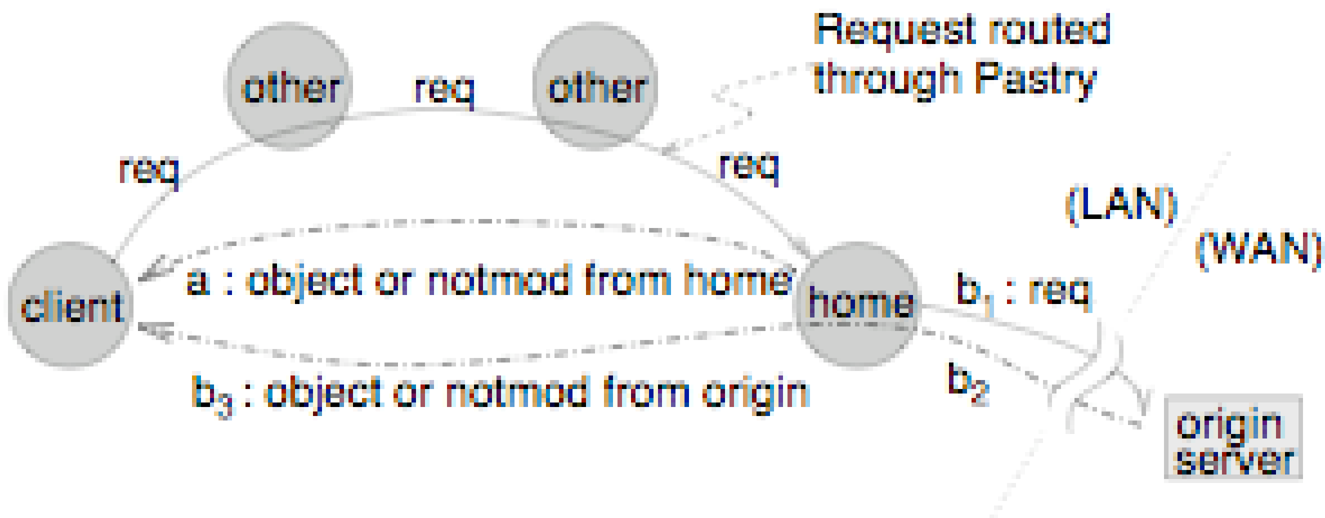
Pastry Routing Example



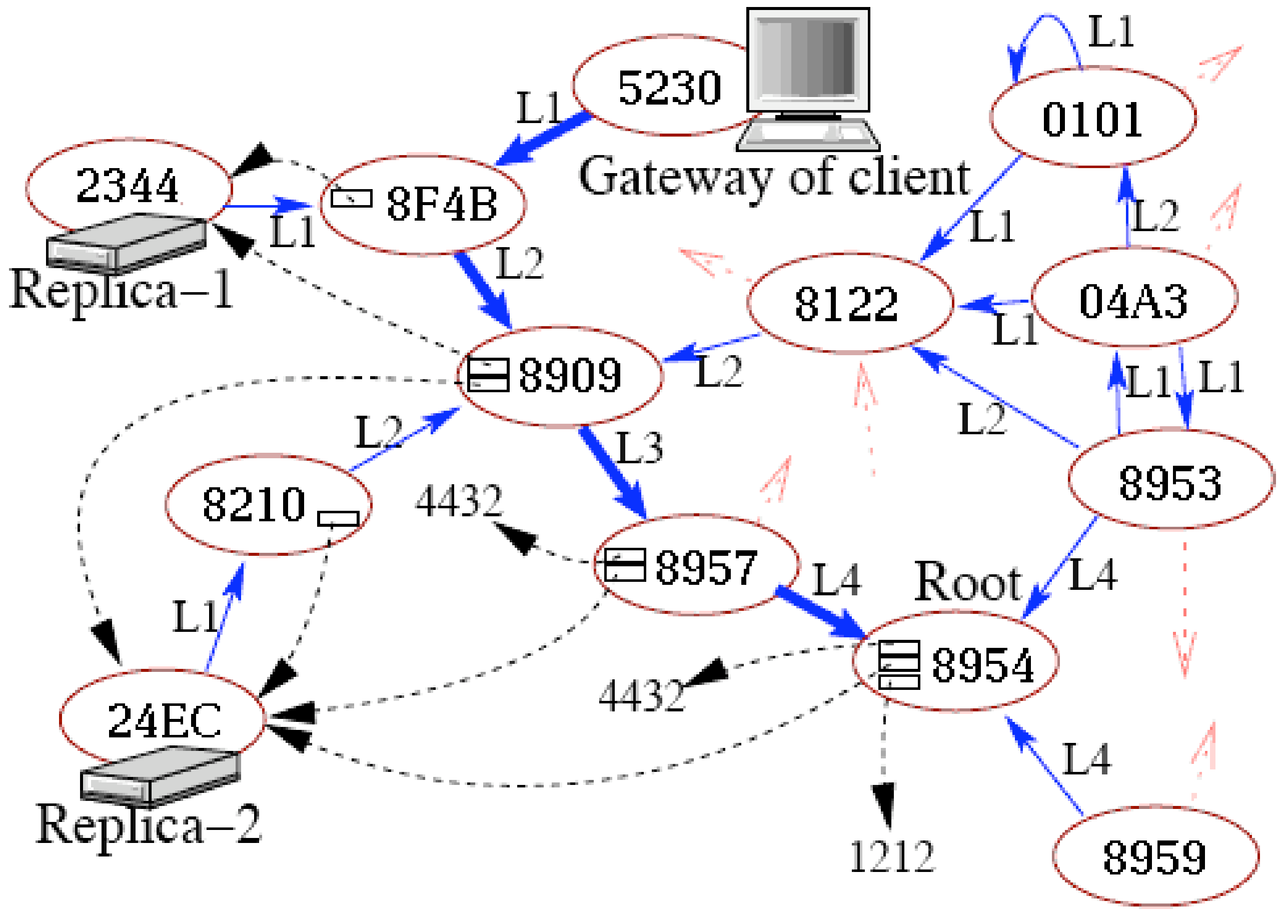
Peer-to-Peer Replication

- Unstructured P2P Background
- Unstructured P2P Replication Strategies
- Structured P2P Background
- Structured P2P Replication Strategies

Squirrel P2P Web Cache



Replication in P2P Systems



Beehive System

- Assumes that queries follows a Zipf distribution
- Assigns replication level using Zipf
- Replicate at all matching peers with prefixes matching replication level
- Average number of hops is constant

Conclusion

- Where to place replicas
  - – Unstructured has difficulty of discovery
  - – Structured has difficulty of locality
- Peers can show strong locality
  - – How to proactively place replicas
  - – Try guessing next file to place based on past