YAPPERS & ZIGZAG

Papers on two different Peer-to-Peer Methodologies

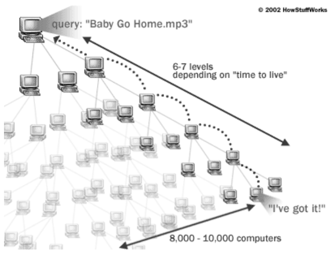
Review by Dave Tucker
Paper Code: P2P – 1
9/30/2003

YAPPERS

- Authors:Prasanna Ganesan, Qixiang Sun, Hector Garcia-Molina; All of CS dept at Stanford
- What is YAPPERS?
  - Yet Another Peer-to-Peer System
  - A peer-to-peer lookup service over arbitrary topology.
  - What is peer-to-peer?
    - Peer-to-peer systems and applications are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality.
    - The core operation in most peer-to-peer systems is efficient location of data items.

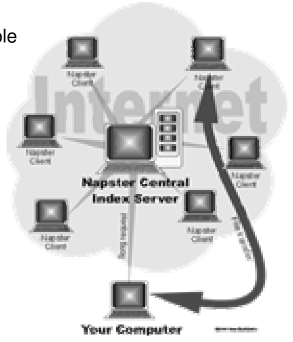
Peer-to-Peer

- Gnutella Style:


© 2002 HowStuffWorks
query: "Baby Go Home.mp3"
6-7 levels depending on "time to live"
"I've got it!"
8,000 - 10,000 computers

Peer-to-Peer

- Distributed Hash Table (Napster style):



YAPPERS

- Why do we need another Protocol?
  - Problems with existing protocols like Gnutella-style
    - Are inefficient
    - Do not search all the nodes
  - Problems with Distributed Hash Table protocols
    - No distinction between a full and partial lookup and therefore Can't take advantage of a partial lookup
    - "Hot Spot" Problem – Popular searches create a bottle neck
    - Have to impose strict constraints on the network topology
    - Additional nodes are a problem. - all hash tables must be updated

YAPPERS

- Problems YAPPERS solves?
  - Best of both worlds (DHT and Gnutella-style) & more
    - Efficiency of a Distributed Hash Table lookup (instead of flooding blindly)
    - Do not have to have one central Hash Table like most DHT (like Napster style)
    - Ability to do a total lookup of all nodes
    - Ability of partial lookups (subset of requested query)
    - Contact only nodes that may contribute to the query
    - Minimize effect of topology changes
    - No constraints on topology

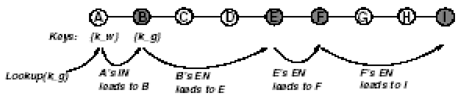
YAPPERS

- Overview
  - Keyspace is partitioned into a small number of buckets
  - Article uses simple example using colors as a bucket to explain.
    - Gray and White – all nodes are either gray or white depending on their data being stored
    - A query for a gray key needs only be done on all the gray keys.
    - What if query originated on a White node?
      - Query forwarded to a neighboring gray node.
      - If no neighboring gray nodes then need to expand neighborhood and assign multiple colors to one node.

7

YAPPERS

- The gray query is forwarded only to other gray nodes.
- Thus avoiding searching any of the white nodes. Thereby cutting down the # of nodes that need to be searched.
- Require that each node knows of all other nodes within 3 hops of the original node.
  - Will forward the query to these nodes
  - The query terminates only after all nodes have been reached or partial lookup number reached



8

YAPPERS Basic Algorithm

- The protocol divides a large overlay network onto many small overlapping neighborhoods
  - Neighborhoods being immediate neighborhoods
    - Immediate neighborhoods are nodes that are 3 or less hops away - # of hops may actually vary
  - When a lookup/query occurs that the local neighborhood does not satisfy then the query is forwarded to all the immediate neighborhoods.
  - If still no answer these will continue to their immediate neighborhoods.

9

YAPPERS – Hashing

- Where the Hashing comes into play
  - YAPPERS partitions the key space among the nodes of a given neighborhood using hash values of the IP addresses.
    - HASH(k) ≡ (HASH(IPx) mod b)
      - b is number of HASH buckets
      - k is the key
    - Entire keyspace is divided into b hash buckets (b different colors)
  - This Hashing is done by the node itself and will determine which bucket/color k is. Then notifies every node in the neighborhood of that same color of the (k,v) relation on it's computer

10

YAPPERS – Total Search

- What if we want to search the entire network effeciently
  - Want to avoid looking at nodes that we know will not have the result of the query (Gnutella)
  - Have every node keep track of a bigger, extended neighborhood so it can make bigger jumps
  - Extended neighborhood is the collection of all nodes in the immediate neighborhood and all of the immediate neighborhoods of those nodes.

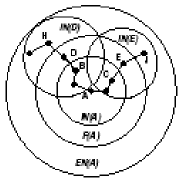


Fig. 2. The extended neighborhood of node A, $EN(A)$, is the union of the immediate neighborhood of nodes in the frontier of A.

11

YAPPERS – Total Search

- Search starts at A
- Looks in it's IN for results, they return results and those nodes that were queried continue the search.
- These are still only querying nodes of the same color so to avoid searching nodes that will not have the results of the query.
- To avoid cycles when forwarding, a unique identifier is assigned to the query and can be used to break the searching.
- Article then goes on with formal proofs

12

YAPPERS – Maintaining Topology

- Node Leaves
  - Broadcast the deletion to a field of 2(h) hops, example h=3 or 6 hops.
  - Each YAPPERS node updates the topology and adjusts it's immediate and extended neighborhood accordingly.
  - Each still existing node in the neighborhood will have to find another node to re-add it's (key, value) to that were previously stored on the deleted node.

13

YAPPERS – Maintaining topology

- Node Enters
  - Must readjust IN and EN information
  - But also keep in mind that just appending node information is inefficient and doesn't maintain the YAPPERS topology.
  - The mapping must be trimmed to reflect only the IN and EN relationships
- Advantage of YAPPERS to some of the other protocols is that node deletion and insertion is independent of the rest of the network, and only the local IN and EN are involved.

14

YAPPERS – Disadvantages

- Disadvantages Mentioned in the Article
  - What if multiple nodes in an immediate neighborhood have the same color as the key (redundant searching)
  - No nodes in the immediate neighborhood have the same color as the key (Won't search that neighborhood)
    - Solution – assign a node in the immediate neighborhood that color as a secondary color then continue.

15

YAPPERS – Summary

- Combines the features of Gnutella/unstructured and DHT types of P2P protocols.
- Partial and Total lookups
- "Filtered" queries

16

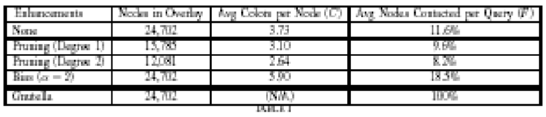
YAPPERS - Enhancements

- Address the problem of highly variable node degrees
  - Fringe node problem
    - Many secondary nodes
  - Fan-out problem
    - Good for partial look ups
    - Bad for total look ups
  - Paper comes up with solutions

17

Actual Experiment

- Used a Gnutella network or 24702 nodes
  - Ran total lookup query
  - Number of Hops = 2
  - Compared search efficiency and Search overhead
  - Varied the number of buckets (colors)
  - Comments on Results:
    - Like to see actual time results, if only a 5ms difference is it worth the additional complexities of this protocol?
    - Could be biased – with proper selection of "colors"



Enhancements	Nodes in Overlay	Avg Colors per Node (C)	Avg Nodes Contacted per Query (F)
None	24,702	3.73	11.6%
Pruning (Degree 1)	15,785	3.10	9.6%
Pruning (Degree 2)	12,081	2.64	8.2%
Bins (α = 2)	24,702	5.90	18.5%
Gnutella	24,702	(N/A)	100%

TABLE I
SEARCH EFFICIENCY OF RUNNING YAPPERS WITH b = 32 ON GNUTELLA SNAPSHOT AND VARIOUS ENHANCEMENTS

18

YAPPERS – Evaluation/Conclusion

- Strengths of Paper
  - – Good overall layout
  - – Does overview of each major topic before going into detail
  - – Addresses problems you would commonly think of with a P2P network
  - – Volunteering & addresses Pitfalls
    - Comes up with fixes
  - – Admits to problems, even if there is no solution

19

YAPPERS – Evaluation/Conclusion

- Weakness of Paper
  - – Like to see real-life examples instead of “colors” and generic search of data.
  - – Maybe more emphasis on the topic of maintaining topology, as this is very important with P2P networks, especially when they will run on the internet
    - To their credit they address this a little at the end
  - – Some hand waving as far as how to transfer queries to other neighborhoods.
  - – Article assumes there is a third-party network layer that determines the live nodes and the neighborhood when a new node is created.
  - – Experiment didn’t compare to a DHT

20

ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming

- What is ZIGZAG?
  - – Peer-to-Peer protocol for single source media streaming to multiple receivers.
  - – Uses a tree structure
- What problem is it solving?
  - – This protocol can maintain the stream under the effects of network dynamics and unpredictable client behavior.
  - – i.e. If failure then it can have a quick and graceful recovery.
  - – Shorten delay for source to receivers.
  - – Limit the required overhead on the receivers.
  - – Realize that this basic structure is somewhat common but zigzag connects them differently.

21

ZIGZAG – How it works to solve the problem

- Organizes the nodes in two different ways.
  - – 1. Administrative
    - Organized into a tree of clusters.
    - Each cluster having at least 4 nodes (except top)
      - – One of those nodes will be the head of the cluster
      - – Head node of a cluster becomes a member of parent cluster

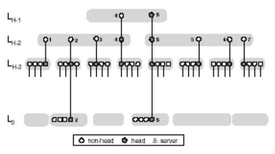


Fig. 1. Administrative organization of peers

22

ZIGZAG – How it works to solve the problem

- – 2. Multicast Tree
  - Join, Departure, & Optimization must follow rules
  - Rules for peer connections are demonstrated below
    - – Peer, when not at its highest layer cannot have any like to or from any other peer
    - – A peer when at its highest layer can only link to its foreign subordinates
    - – At layer j<H-1 (below top 2) get content from foreign head

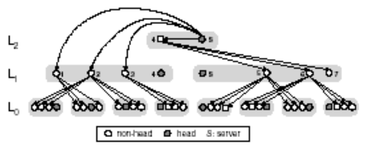


Fig. 2. The multicast tree of peers ($H = 3$, $k = 4$)

23

ZIGZAG – How it works to solve the problem

- – Control Protocol
  - Used to maintain connections & position
  - Each node in a cluster periodically communicates with its clustermates, children and parent on the tree.
    - – Which nodes it is communicating with
    - – Which clusters have room for another node
    - – Which nodes are have paths to the lowest level of the tree
- – Node Enters
  - When a peer sends a request to the server
  - Push request down the multicast tree until a leaf node is found that has room in it’s cluster
  - Periodic check of the clusters to see if they are to big, if so split them

24

ZIGZAG – How it works to solve the problem

- Nodes Leaves
  - When a peer departs from the tree
  - Parent, subordinates and children of the node are notified
  - Non-head clustermate is selected and asked to take over forwarding data
  - If many departures causes undersize then 2 cluster at the same layer are merged

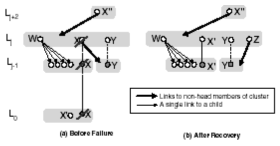


Fig. 4. Failure Recovery: Peer X fails

25

ZIGZAG – How it works to solve the problem

- Performance Optimization
  - Periodically checking the administrative organization and multicast tree.
  - If node has many children may consider switching parenthood of some of the children to a clustermate - Balancing based on number of nodes
  - Switching based on capacity or bandwidth – Balancing based on bandwidth

26

ZIGZAG – Experiment

Look at various scenarios on a network simulator running 3240 nodes,2000 client.

- 5 – 15 nodes per cluster.
- Situations with no failure and adding clients
  - Looked at overhead such as adding new nodes average of 2.4% population were contacted
  - If too many children then it must ask most of them
  - Performance improved when a split of a cluster occurred
- Situations where Failure occurs
  - Let every 200 nodes fail (out of 2000) sequentially.
  - Mostly less than 20 reconnections (2% of population)

27

ZIGZAG – Experiment

- ZIGZAG vs. NICE
- NICE – Recently developed protocol for multimedia streaming, also organizes nodes in clusters

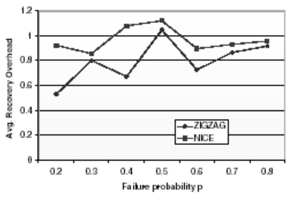


Fig. 9. ZIGZAG vs. NICE: Failure Recovery Overhead

28

ZIGZAG – Author Advantages/Disadvantages

- Advantages of this approach
  - Oddities in the overhead of streaming multimedia over the internet will not felt as much by the end user.
  - Efficient methods for nodes dropping & adding
    - Uses two trees
      - 1 to maintain administration
      - 1 for data flow
  - Short end-to-end delay
    - Tree structure
  - Low administration overhead
    - Nodes “talk” to each other on regular basis

29

ZIGZAG – Author Advantages/Disadvantages

- Disadvantages of this approach
  - Possible for a lot of overhead per node and especially head nodes
  - Needs to Optimize at random intervals to be effective which may cause large overhead.
  - Author didn’t address much.

30

ZIGZAG – Paper Evaluation

- Strengths
  - Compares to other systems
  - Runs through several scenarios when testing
  - Close comparison to another similar system
- Weaknesses
  - Would like to see actual test vs. on a network simulator
  - Lot of nomenclature that is confusing
  - Difficult to follow at times.
  - Needs an example to help explain how it works
  - Author didn't address any disadvantages directly

Questions

- Explain one improvement of Yappers over Gnutella style of P2P search.
- Briefly, how does Yappers Use both, the Gnutella and Distributed Hash Table approach to resolving a query.
- In the Yappers Article what's the difference between a partial lookup and total lookup?
- What simple approach does ZIGZAG do to handle administration of nodes and data delivery.
- What problem(s) does ZIGZAG solve?