Probabilistic Location and Routing

IEEE INFOCOM 2002

謝志峰

2003/7/23

Outline

- Introduction
- Bloom Filters
- Attenuated Bloom Filters
- Tapestry
- Simulation
- Conclusions

Introduction (1/2)

- It introduces two important challenges to system architects.
  - First, if replicas may be placed anywhere, how should we locate them?
  - Second, once located one or more replicas, how should we route queries to them?

Introduction (2/2)

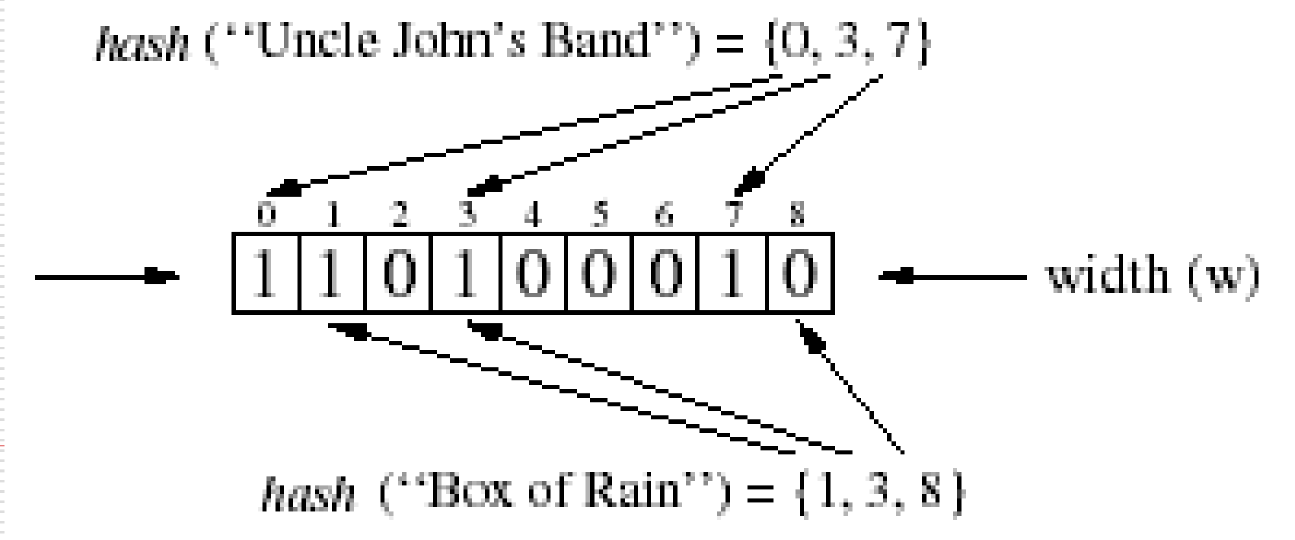
- Probabilistic location and routing algorithm is based on attenuated Bloom filters:
  - It is decentralized.
  - It is locality aware.
  - It follows a minimal search path.
  - It uses constant storage per server.

Bloom Filters(1/2)

- A Bloom filter is a bit-array of length w with independent hash functions.
- To determine whether contains a given element:
  - If any of the bits are not set, the represented set definitely does not contain the object.
  - If all of the bits are set, the set may contain the object.

Bloom Filters(2/2)

- An array of W bits that serve to summarize a set of objects.
- The represented set probably contains the name "Uncle John's Band", since bits 0, 3, and 7 are all true.
- It definitely does not contain "Box of Rain", since bit 8 is false.

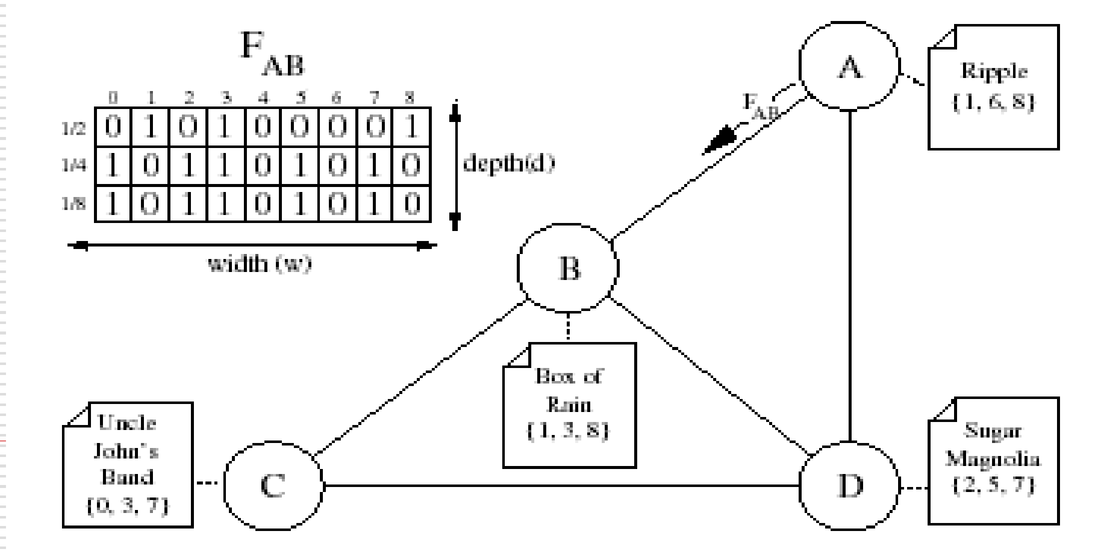


Attenuated Bloom Filters (1/4)

- An attenuated Bloom filter of depth d is an array of d normal Bloom filters.
- The ith Bloom filter is the merger of all Bloom filters for all of the nodes a distance i through any path starting with that neighbor link.
- The distance is in terms of hops in the overlay network.

Attenuated Bloom Filters (2/4)

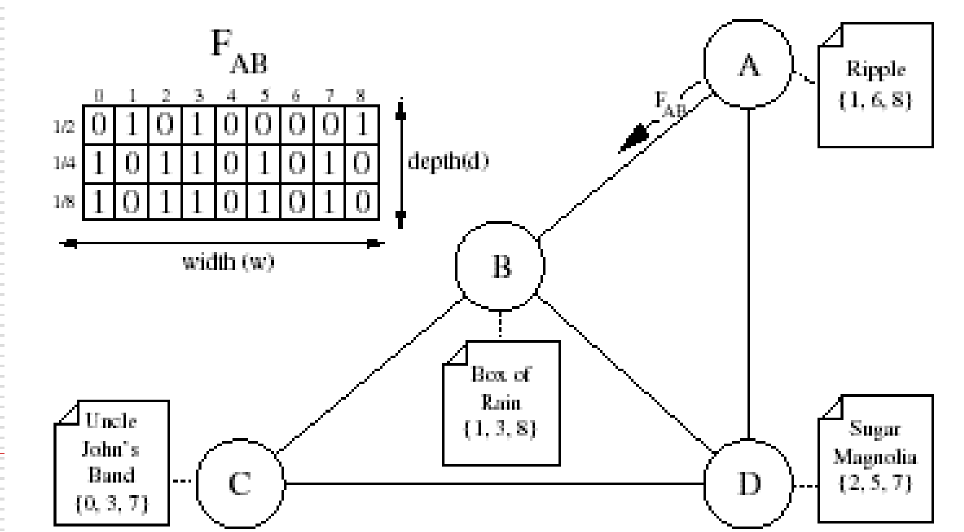
- Filters are labeled with their level (top filter is level 1). Each outgoing link (say, A -> B) with it (F_{AB}).
- Level 1 summarizes replicas on the neighbor at the end of the link.
- Level 2 summarizes replicas that are two-hops away along that link, etc.



Attenuated Bloom Filters (3/4)

- Both "Uncle John's Band" and "Sugar Magnolia" are two hops away from Node A through Node B, so the second level of filter F_{AB} contains true values (0, 2, 3, 5, 7).

- In F_{AB} , the document "Uncle John's Band" would map to the potential value 1/4+1/8=3/8



Attenuated Bloom Filters (4/4)

- For example

  If potential value

  =13/32

  =8/32+4/32+1/32

  =1/4+1/8+1/32

  Location in **L2,L3,L5**

L1	1/2
L2	1/4
L3	1/8
L4	1/16
L5	1/32

1	0	0	1	0	1	1	0
1	0	1	0	1	0	1	1
0	1	1	1	1	0	1	1
1	1	0	1	1	1	0	1
0	0	1	0	1	0	1	0

The Query Algorithm

- To perform a location query, the querying node examines the 1st level of each of its neighbors filters.
  - If one matches , query is forwarded to the matching neighbor closest to the current node.
  - If no filter matches, the querying node looks for a match in the 2nd level of every filter.
  - The query can be returned to be sent on to the next best neighbor.

The Update Algorithm (1/2)

- When a new document is stored, the server calculates the changed bits in its own filter and in each of the filters its neighbors maintain of it.
  - It then sends these bits out to each neighbor.
  - On receiving messages, each neighbor attenuates the bits one level and computes the changes they will make in each of its own neighbors' filters.

The Update Algorithm (2/2)

- We then perform the following types of filtering:
  - destination filtering: Destination servers remember the identifiers of every update they see. This filtering prevents redundant information.
  - source filtering: Once receiving a duplicate update, it sends a message to that neighbor to inform it of this redundancy and stops forwarding new updates.

Tapestry (1/4)

- Tapestry begins with the assumption that
  - Every server and document can be named with a unique, location independent identifier.
  - Node-IDs for the node names and globally unique identifiers (**GUID**s) for the documents.
  - It represented as a sequence of hexidecimal digits.
- A query is routed from node to node until the location of a replica is discovered, at which point the query proceeds to that replica.
- Once Tapestry has discovered the location of a replica, it forwards the query to the replica closest to the point of discovery.

Tapestry (2/4)

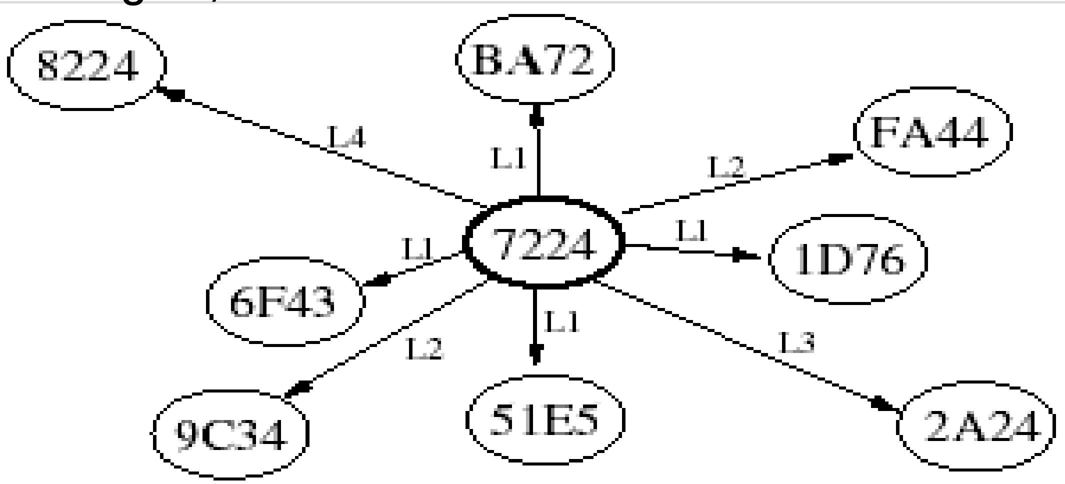
- Every node is connected to other nodes via neighbor links of various levels.
  - Level-1 edges from a node connect to the 15 nodes closest with different values in the lowest digit of their addresses.
  - Level-2 edges that match in the lowest digit and have different second digits, *etc.*

7224--> L1-->BA72

722**4**--> L2-->FA4**4**

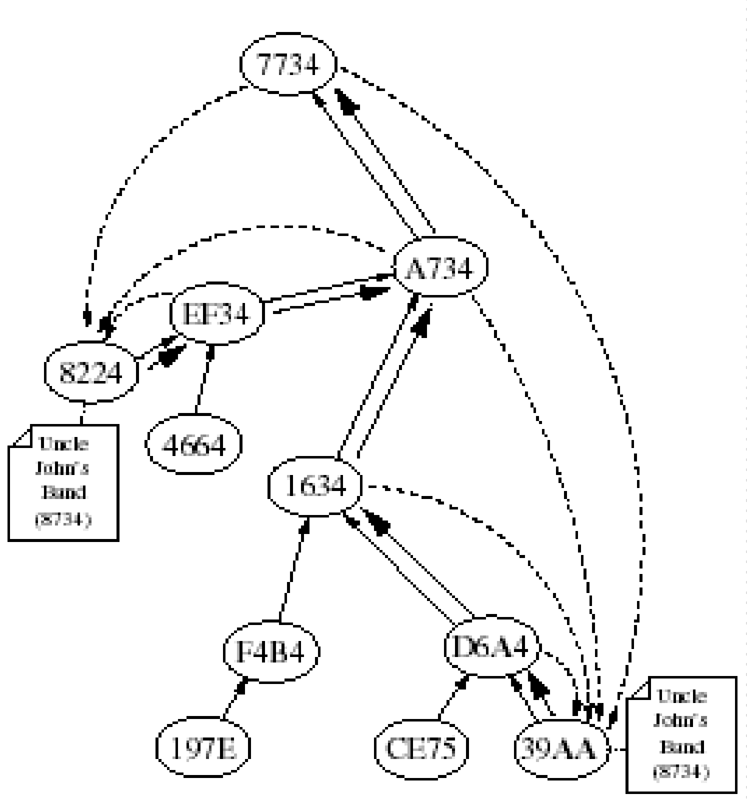
72**24**--> L3-->2A**24**

7**224**--> L4-->8**224**



Tapestry (3/4)

- Publication in Tapestry.

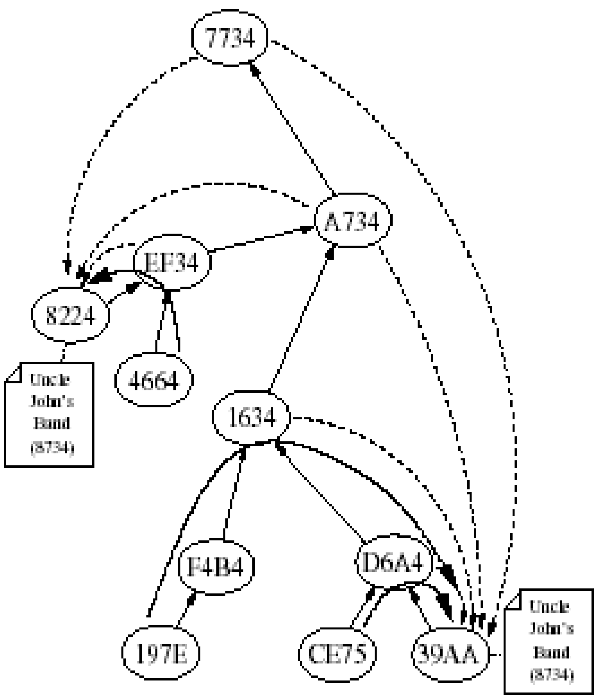


- It illustrates two replicas with the same GUID (8734) exported by server nodes 8224 and 39AA.

- To publish document 8734, server 39AA sends publication request towards the root.

Tapestry (4/4)

- Queries route toward the root node until they encounter a location pointer, then route to the located replica.
- If multiple pointers are encountered, the query proceeds to the closest replica.
- In the worst case, a location operation involves routing all the way to the root.



Simulation (1/4)

- We constructed a physical network topology using the transit-stub model of GT-ITM [16].

  - all stub to stub edges are 100 Mb/s
  - all stub to transit edges are 1.5 Mb/s
  - all transit to transit edges are 45 Mb/s.

    (Fast Ethernet, T1, and T3 connections).

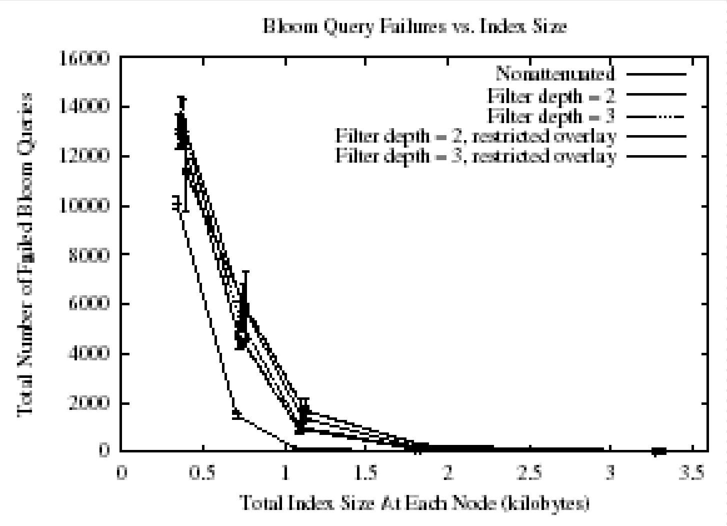
  - The **static** and the **dynamic** experiments are based on whether the set of replicas changes during the test.

Simulation (2/4)

- As the width of the bloom filters increases, the false positive rate drops quickly.
- A total index size of around 1.83 kilobytes is sufficient to limit the number of such failing queries.

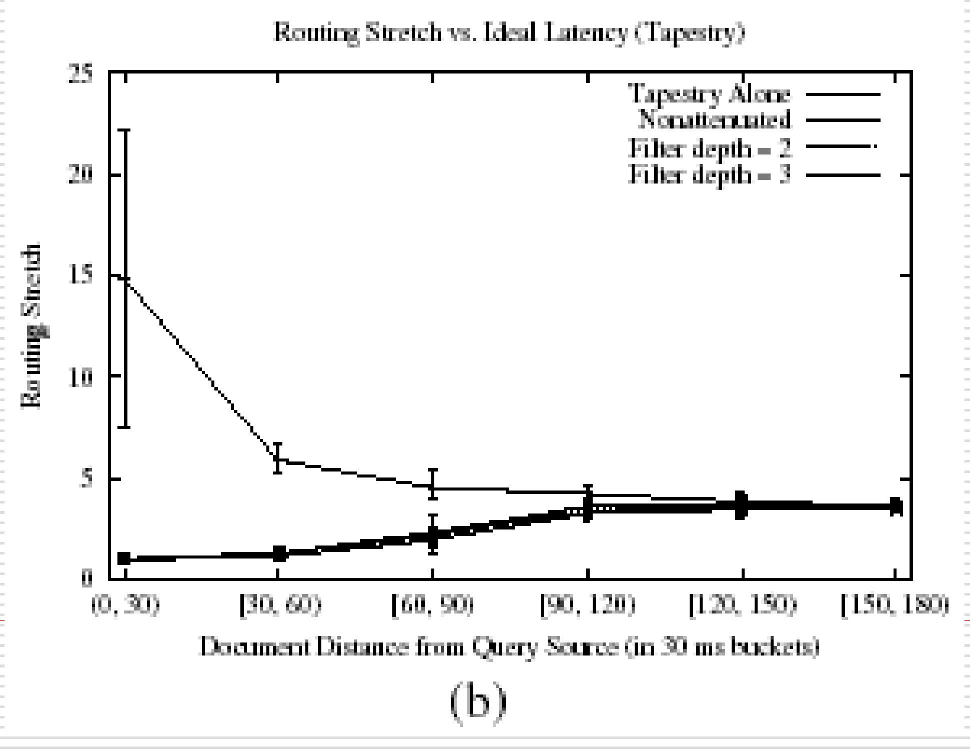
- Static Experiment

Bloom Query Failures vs. Index Size.



Simulation (3/4)

- The total size of the Bloom filter index at each node is fixed at 0.136 percent of the data size, as suggested by the previous results.

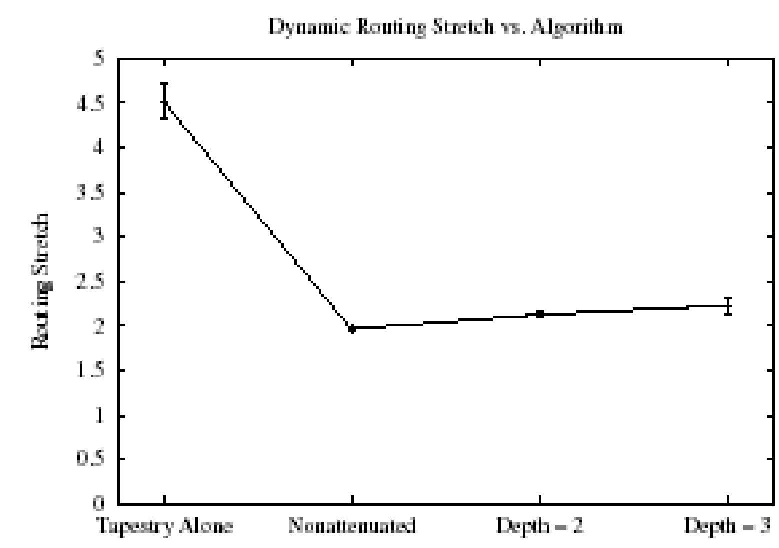


Simulation (4/4)

- This graph shows the average routing stretch as a function of routing algorithm for the dynamic simulations.

- The hybrid algorithm far outperforms Tapestry alone for all filter depths.

- *Dynamic Routing Stretch vs. Algorithm.*



Conclusions

- probabilistic routing algorithm designed to improve the location latency of existing deterministic approaches.
- The algorithm finds nearby replicas quickly, and if no such replicas exist, it fails quickly as well.
- The algorithm may be combined with a deterministic algorithm to improve average routing stretch for nearby documents.

Discussions

- Cache Usage
- Attenuated bloom filter
  - The array must be very large in large network.
- Tapestry
  - Roots must have large memory.
  - Roots look like servers.
  - Query for the top of root last time
  - Tree