

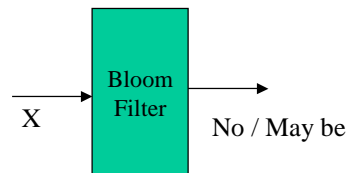
Review of:

Network Applications of Bloom Filters : A Survey

- Paper by:
 - Andrei Broder and Michale Mitzenmacher
- Presented at:
 - To be published
- Survey by:
 - Sarang Dharmapurikar

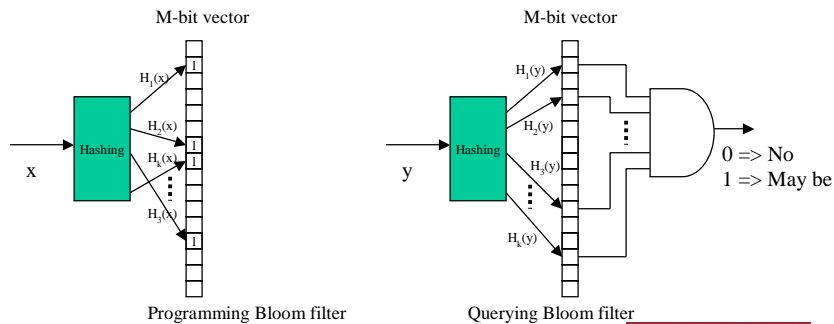
Basic Idea

- Querying a data base for membership
 - Given a data base of n messages, find out if a message x belongs to the data base
- Bloom filter gives one of the following answers
 - No, x does not belong to the data base of n messages (100% confidence)
 - x may belong to the data base of n messages (ambiguity)



Technique

- Programming Bloom filter
 - Given a message x , obtain k addresses by performing k hash functions on x
 - Set the bit at each address in a m -bit long vector
 - Do the same for each of the n messages
- Querying Bloom filter
 - Given a message y , obtain k addresses by performing the same k hash functions on y
 - Read the bit at each address from the same M -bit long vector
 - AND all the k bits read
 - 0 => the message not in the data base
 - If it was then all the bits would be definitely set
 - 1 => the message "may" be in the data base
 - "May be" because the bits could have been set by some other messages too



Math

- The probability that a hash function sets a bit at a random address A is $1/m$
- The probability that a hash function does not set the bit at address A is $1 - 1/m$
- The probability that the k successive hash functions generated for a message don't set the bit at address A is $(1 - 1/m)^k$
- Since each of the n messages generates k hash functions, the probability that the bit at address A is not set by any of these hash functions is $(1 - 1/m)^{nk} \approx e^{-kn/m}$
- The probability that the bit is set by some message is $1 - e^{-kn/m}$
- The probability that k such bits generated by hash functions on a single message are all set is $[1 - e^{-kn/m}]^k = \text{probability of false positive}$

Counting Bloom filters

- How to program Bloom filter with more messages on the fly?
 - Easy, the same technique, k hash functions set k bits in the filter
- How to remove some messages (rules) from the Bloom filter?
 - The k bits set during the programming for the rule to be removed can not be reset, some other rule might have set some of the bits
- Instead of single bit vector maintain a vector of counters as Bloom Filter
 - All the counters are reset in the beginning
 - For programming, increment the k counters corresponding to the k hash values on a rule
 - When a rule is to be removed, decrement the counters that were incremented by the rule
 - This ensures that the other rules are not disturbed

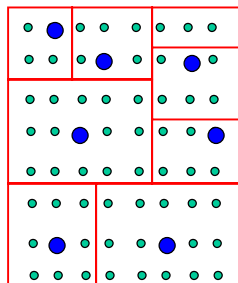
Applications

- Collaborating in overlay and peer-to-peer networks
 - Bloom filters can be used for summarizing content to aid collaborations in overlay and peer-to-peer networks
- Resource Routing
 - Bloom filters allow probabilistic algorithms for locating resources
- Packet routing
 - Bloom filters provide means to speed up or simplify packet routing protocol
- Measurement
 - Bloom filters provide a useful tool for measurement infrastructures used to create data summaries in routers or other network devices

Distributed Caching

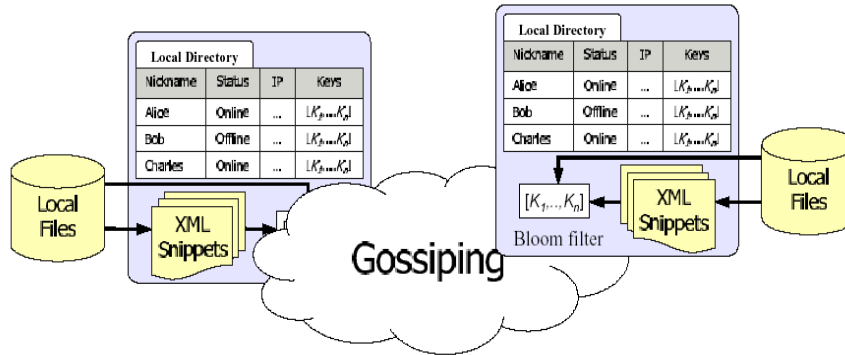
- For fetching the contents of a web page proxies are looked up first
- On cache miss, a proxy attempts to determine if another proxy cache holds the desired web page
- Hence the proxies need to know each others' contents
- The proxies can exchange the list of URLs but it causes too much message traffic
- Instead proxies pass Bloom Filters of the URLs, which are much more compact
- A false positive will cause a request to be made to a proxy which doesn't have the requested web page, but false positives are rare
- Cache contents in the proxies changes hence maintain a counting Bloom Filter

P-2-P networks example : CAN



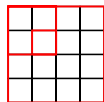
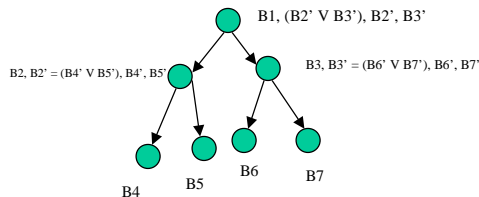
- Every node in the CAN has a coordinate
- Every object to be stored has a unique key
- A deterministic hash function maps a key to a node
- The object corresponding to the key is stored(retrieved) at(from) that node
- The entire network is divided into “zones”
- Each zone has a “owner” which maintains the list of (key,value) pair
- When any node in the in the system makes a request for key, the same hash function tells the coordinates of the node which owns it
- The request is routed through the owners of the neighboring zones to the appropriate node

Bloom filters for moderate size P-2-P network : PlanetP



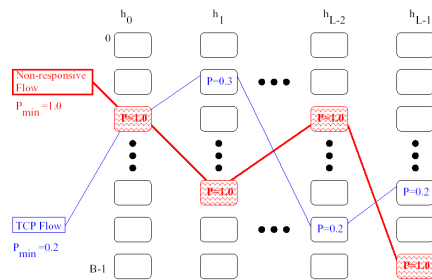
- Replacement for the distributed hashing as in CAN and Chord
- Each peer maintains the list of objects every other peer has
- Usual list data structures have a prohibitively high cost, hence Bloom filters
- A peer maintains some approximate topology of the network
- For retrieving a particular object, key is looked across all the bloom filters and the request is forwarded to the appropriate node through routing table
- Spread the data base by "gossiping"

Resource routing



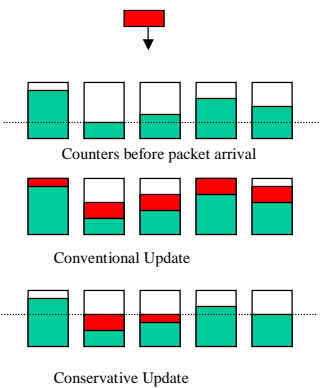
- Arrange the nodes in the network in a tree
- Each node has
 - Bloom filter for its own objects
 - Bloom filters given by the children
 - OR of the Bloom filters given by the children
- When a request for an object is made to a node
 - It queries its own bloom filter
 - If object not found then it queries the OR of the Bloom filters
 - If the object is found then queries the Bloom filters of individual children
 - If the object was not found then the request is routed to the parent node
- Geographic routing
 - Uses the same technique to locate the mobile hosts
 - Divide the region in squares and map the squares to the nodes in a binary tree

Queuing : Stochastic Fair BLUE



- Problem : Some flows hog the bandwidth and do not respond to the TCP congestion notification
- Use Counting Bloom Filters
- L levels (hash functions) each with B counters
- Each of the counter has a drop probability associated with it
- When a packet of a flow arrives, it hashes into L buckets and increments the count
- If the packet count goes above a threshold then drop probability increased by "x"
- If the count is below a threshold then drop probability is decreased by "x"
- If the minimum over all the L probabilities is 1 then rate limit the flow
- Else drop packet with the minimum probability
- Non-responsive flows drive the Pmin quickly to 1

Queuing : Recording Heavy Flows



- Essentially same as SFB with the additional details of flow sampling
- Introduces the idea of the conservative updates

Other applications

- Unix spell checker
- Storing the list of unsuitable passwords for security purposes
- Semi-joins : taking the intersection of the two overlapping distributed databases
- Approximate set reconciliation : taking the union of the two distributed data bases
- IP trace back
 - Each router hashes the packet header into a bloom filter and thus records its arrival
 - By querying the routers for the trace of a particular packet the packet path can be reconstructed
 - False positive results in multiple branches
- Detecting loops
 - Typically TTL is used to detect the loops
 - In small networks TTL takes longer time to detect looping packet
 - Maintain bloom filter in the packet header
 - Each hop sets some bits in the Bloom filter contained in the header
 - If the bits it sets were already set then it can be a loop and hence drop it