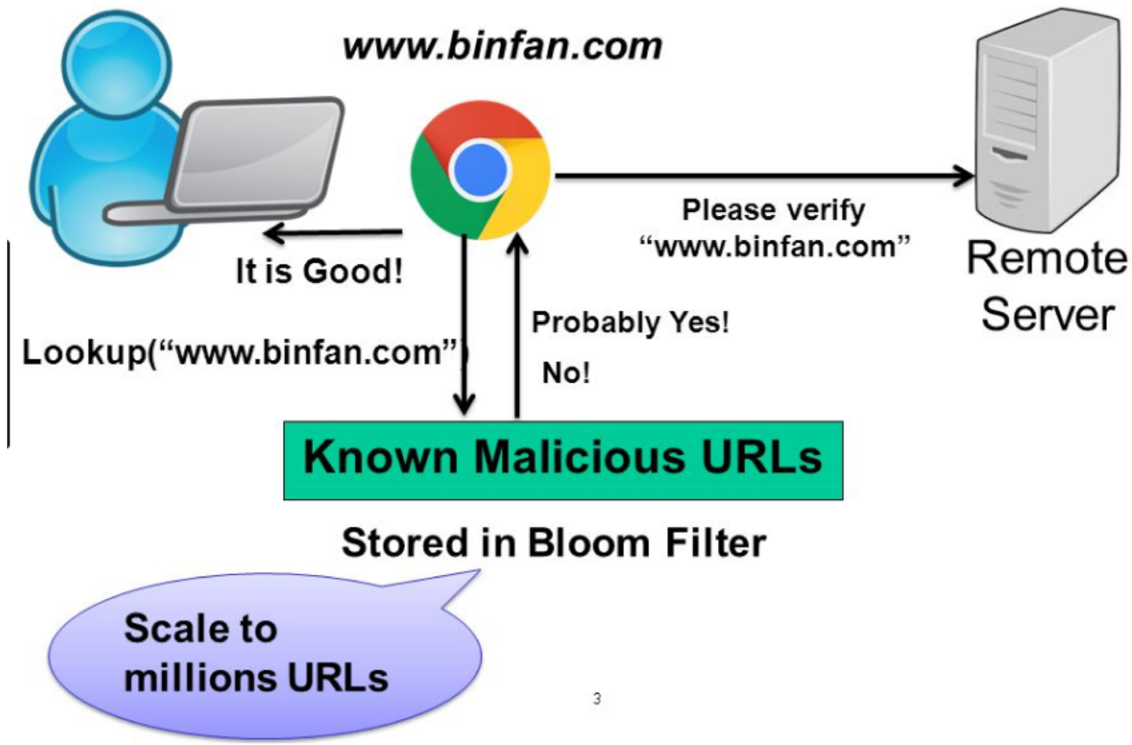
Lecture 10

Bloom’s Filter & Count-Min Sketch

Wilson Rivera

Bloom Filter



Hash Tables vs. Bloom Filter

- BF & HT provide super fast inserts and super fast look ups.
- Boom filter is a lightweight version of hash table with small error probability
- Space efficient (smaller representations)
- Not used to store associate objects
- May be false positives

Bloom Filter

- Given a set $S = \{x_1, x_2, \ldots, x_n\}$, construct data structure to answer queries of the form "Is y in S?"
- Data structure should be:
  - Fast (Faster than searching through S).
  - Small (Smaller than explicit representation).
- To obtain speed and size improvements, allow some probability of error.
  - False positives: $y \notin S$ but we report $y \in S$
  - False negatives: $y \in S$ but we report $y \notin S$

Bloom Filter

- Data set S
- Array A of n bits
- $n/|S|$ = number of bits per object in S
- k hash functions $h_1, h_2, \ldots, h_k$

Bloom Filters

Start with an m bit array, filled with 0s.

B																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hash each item x_j in S k times. If $H_i(x_j) = a$, set $B[a] = 1$.

B																
	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

To check if y is in S, check B at $H_i(y)$. All k values must be 1.

B																
	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

Possible to have a false positive; all k values are 1, but y is not in S.

B																
	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

n items $m = cn$ bits k hash functions

Bloom Filter

Insertion

for i=1,2, …,k

$A[h_i(x)]=1$

Look up

Return TRUE

if and only if

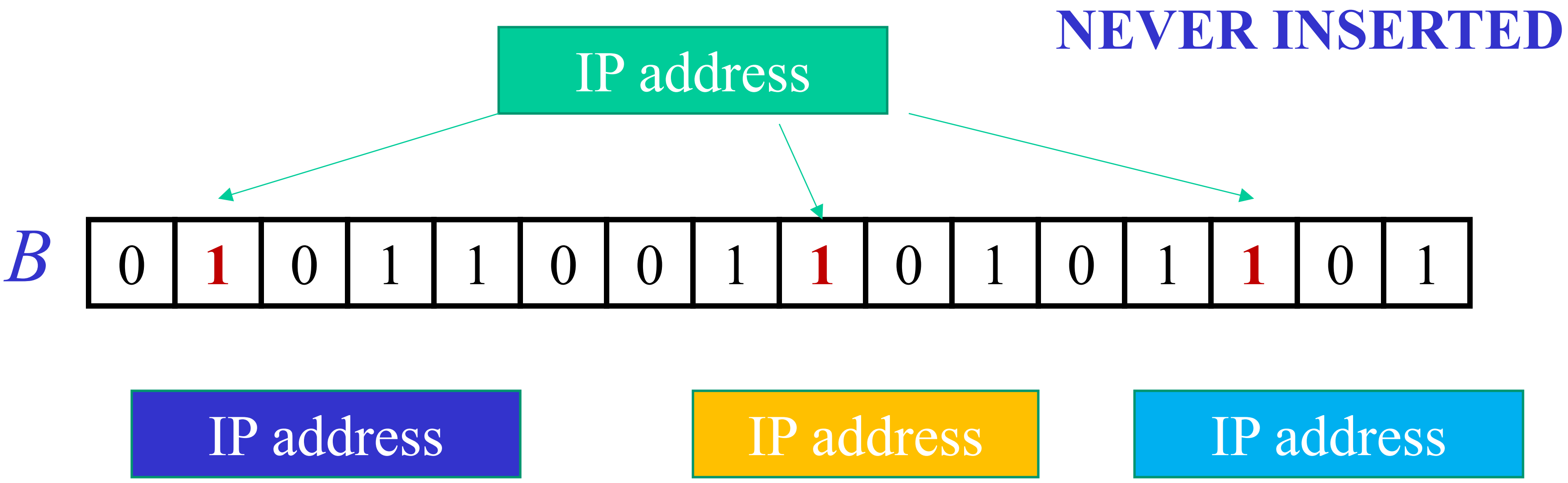
$A[h_i(x)]=1$

for i=1,2, …,k

NO FALSE NEGATIVES

Bloom Filter

- **BUT may be FALSE POSITIVES**



False Positive Probability

- Pr(specific bit of filter is 0) is

$$p' \equiv (1-1/m)^{kn} \approx \mathrm{e}^{-kn/m} \equiv p$$

- If ρ is fraction of 0 bits in the filter then false positive probability is

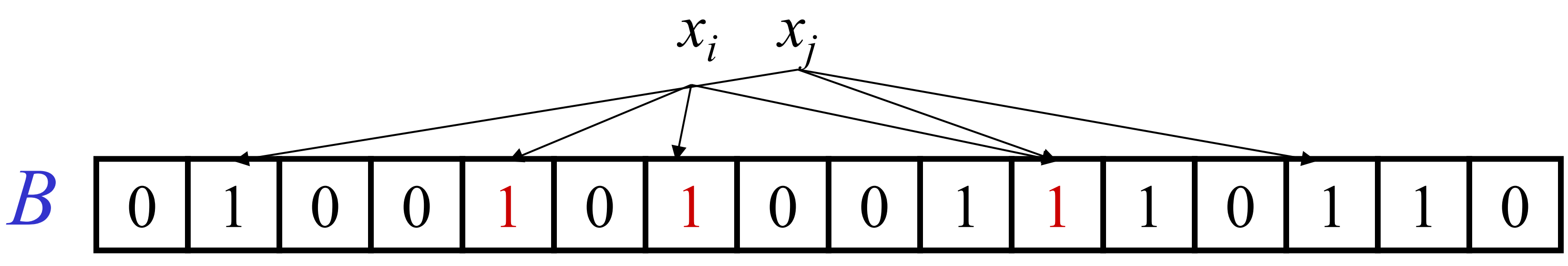
$$(1-\rho)^k \approx (1-p')^k \approx (1-p)^k = (1-\mathrm{e}^{-k/c})^k$$

- Find optimal at $k = (\ln 2)m/n$ by calculus.
  - So optimal fpp is about $(0.6185)^{m/n}$

n items $m = cn$ bits k hash functions

Handling Deletions

- Bloom filters can handle insertions, but not deletions.



- If deleting x_i means resetting 1s to 0s, then deleting x_i will "delete" x_j.

Classic Uses of BF: Spell-Checking

- Once upon a time, memory was scarce...
- `/usr/dict/words` -- about 210KB, 25K words
- Use 25 KB Bloom filter
  - 8 bits per word.
  - Optimal 5 hash functions.
- Probability of false positive about 2%
- False positive = accept a misspelled word
- BFs still used to deal with list of words
  - Password security [Spafford 1992], [Manber & Wu, 94]
  - Keyword driven ads in web search engines, etc

P2P Communication

- Efficient P2P keyword searching [Reynolds & Vadhat, 2002].
  - Distributed inverted word index, on top of an overlay network. Multi-word queries.
  - Peer A holds list of document IDs containing Word1, Peer B holds list for Word2.
  - Need intersection, with low communication.
  - A sends B a Bloom filter of document list.
  - B returns possible intersections to A.
  - A checks and returns to user; no false positives in end result.

P2P Collaboration

- Informed Content Delivery
  - [Byers, Considine, Mitzenmacher, & Rost 2002].
  - Delivery of large, encoded content.
    - Redundant encoding.
    - Need a sufficiently large (but not all) number of distinct packets.
  - Peers A and B have lists of encoded packets.
  - Can B send A useful packets?
  - A sends B a Bloom filter; B checks what packets may be useful.
  - False positives: not all useful packets sent
  - Method can be combined with
    - Recoded symbols (XOR of existing packets)
    - Min-wise sampling (determine a-priori which peers are sufficiently different)

Scalable Multicast Forwarding

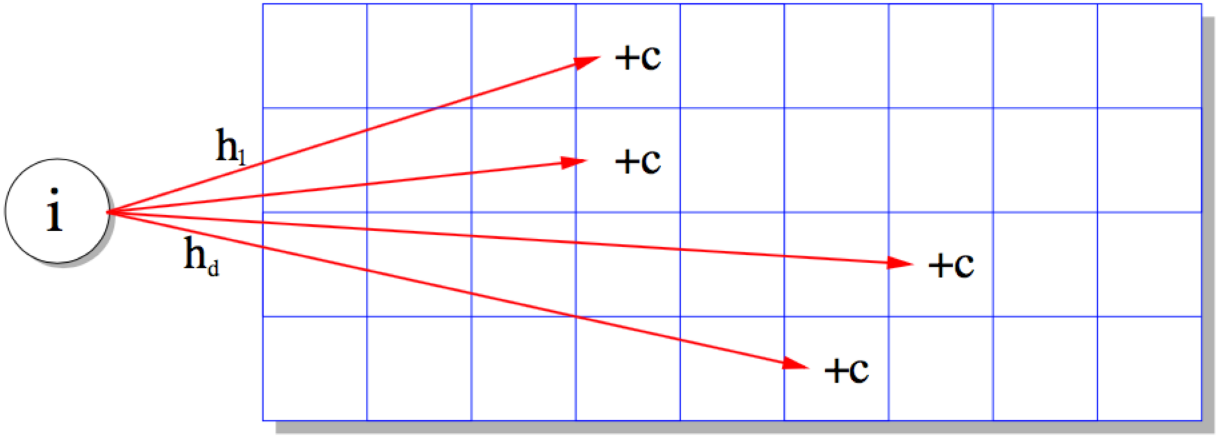
- [Gronvall 02]
- Usual arrangement for multicast trees: for each source address keep list of interfaces where the packet should go
  - For many simultaneous multicasts, substantial storage required
- Alternative idea: trade computation for space:
  - For each interface keep BF of addresses
  - Packets checked against the BF. Check can be parallelized
  - False positives lead to (few) spurious transmissions

Measurement infrastructure

- Hash-based IP traceback
  - [Snoeren et al., 2001]
  - For security purposes, would like routers to keep a list of all packets seen recently.
  - Can trace back path of a bad packet by querying routers back through network.
  - A Bloom filter is good enough to trace back source of a bad packet.
  - False positives require checking some additional network paths.

Count-Min Sketch

Count Tracking, which generalizes membership,



Count-Min Sketch

- The Count-Min Sketch data structure primarily consists of a fixed array of counters, of width w and depth d.
- The counters are initialized to all zeros. Each row of counters is associated with a different hash function.

Count-Min Sketch

- The query HH(k) returns the set of items which have large frequency (say 1/k of the overall frequency).
  - Count tracking can be used to directly answer this query, by considering the frequency of each item.
  - Count-Min sketch can be used in compressed sensing
    - A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. Proceedings of the IEEE, 98(6):937–947, June 2010.

Count-Min Sketch

- Natural Language Processing (NLP)
  – statistics on the frequency of word combinations, such as pairs or triplets of words that occur in sequence.
  – 90GB corpus down to a (memory friendly) 8GB Count-Min sketch
    - Y.-K. Lai and G. T. Byrd. High-throughput sketch update on a low-power stream processor. In Proceedings of the ACM/IEEE symposium on Architecture for networking and communications systems, 2006.