Codes, Bloom Filters, and Overlay Networks

Michael Mitzenmacher

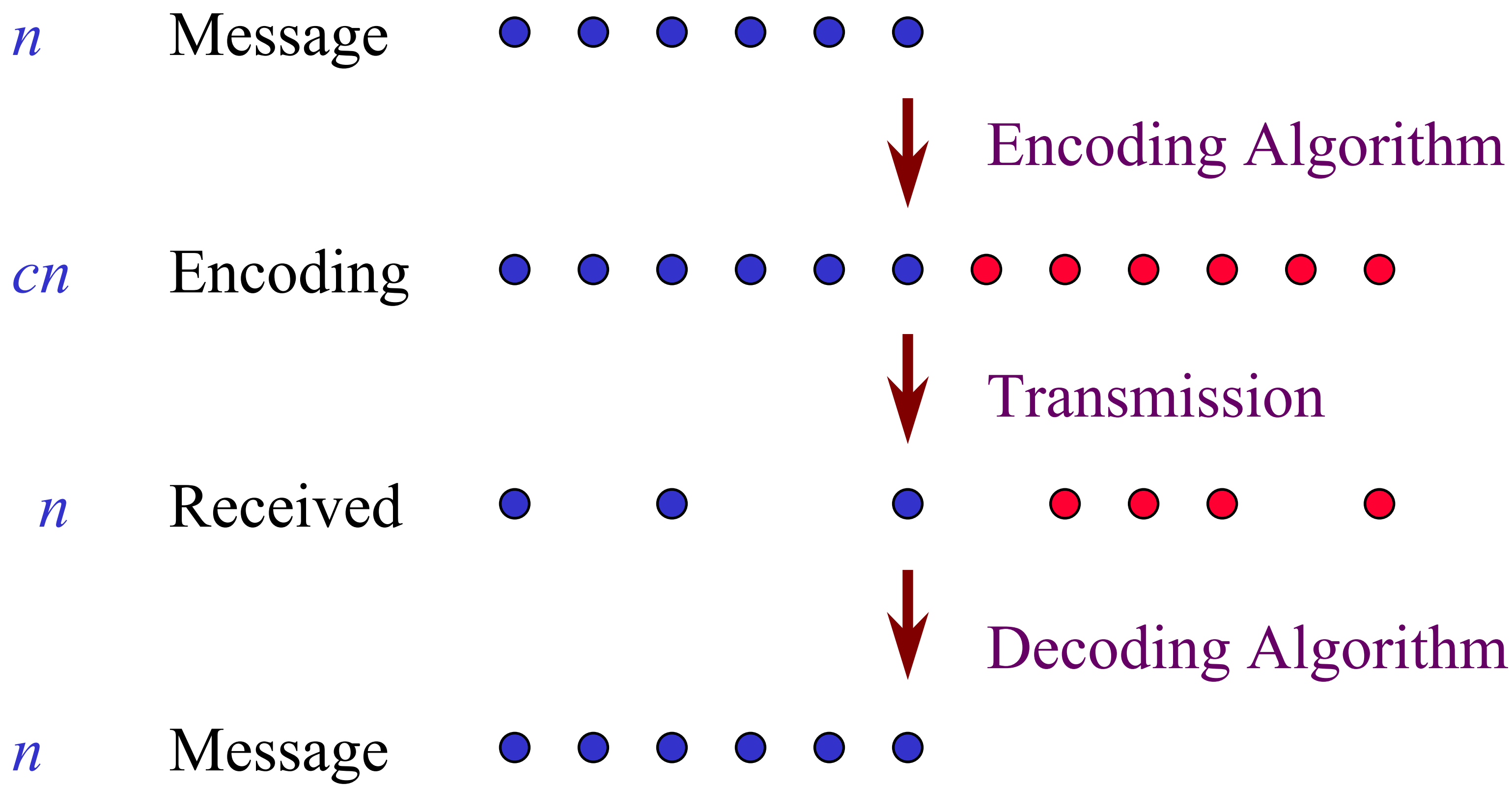
Today...

- Erasure codes
  - Digital Fountain
- Bloom Filters
  - Summary Cache, Compressed Bloom Filters
- Informed Content Delivery
  - Combining the two…
- Other Recent Work

Codes: High Level Idea

- Everyone thinks of data as an ordered stream. *I need packets 1-1,000*.
- Using codes, data is like water:
  - You don't care what drops you get.
  - You don't care if some spills.
  - You just want enough to get through the pipe.
  - *I need 1,000 packets*.

Erasure Codes



Application: Trailer Distribution Problem

- Millions of users want to download a new movie trailer.
- 32 megabyte file, at 56 Kbits/second.
- Download takes around 75 minutes at full speed.

Point-to-Point Solution Features

- Good
  - Users can initiate the download at their discretion.
  - Users can continue download seamlessly after temporary interruption.
  - Moderate packet loss is not a problem.
- Bad
  - High server load.
  - High network load.
  - Doesn’t scale well (without more resources).

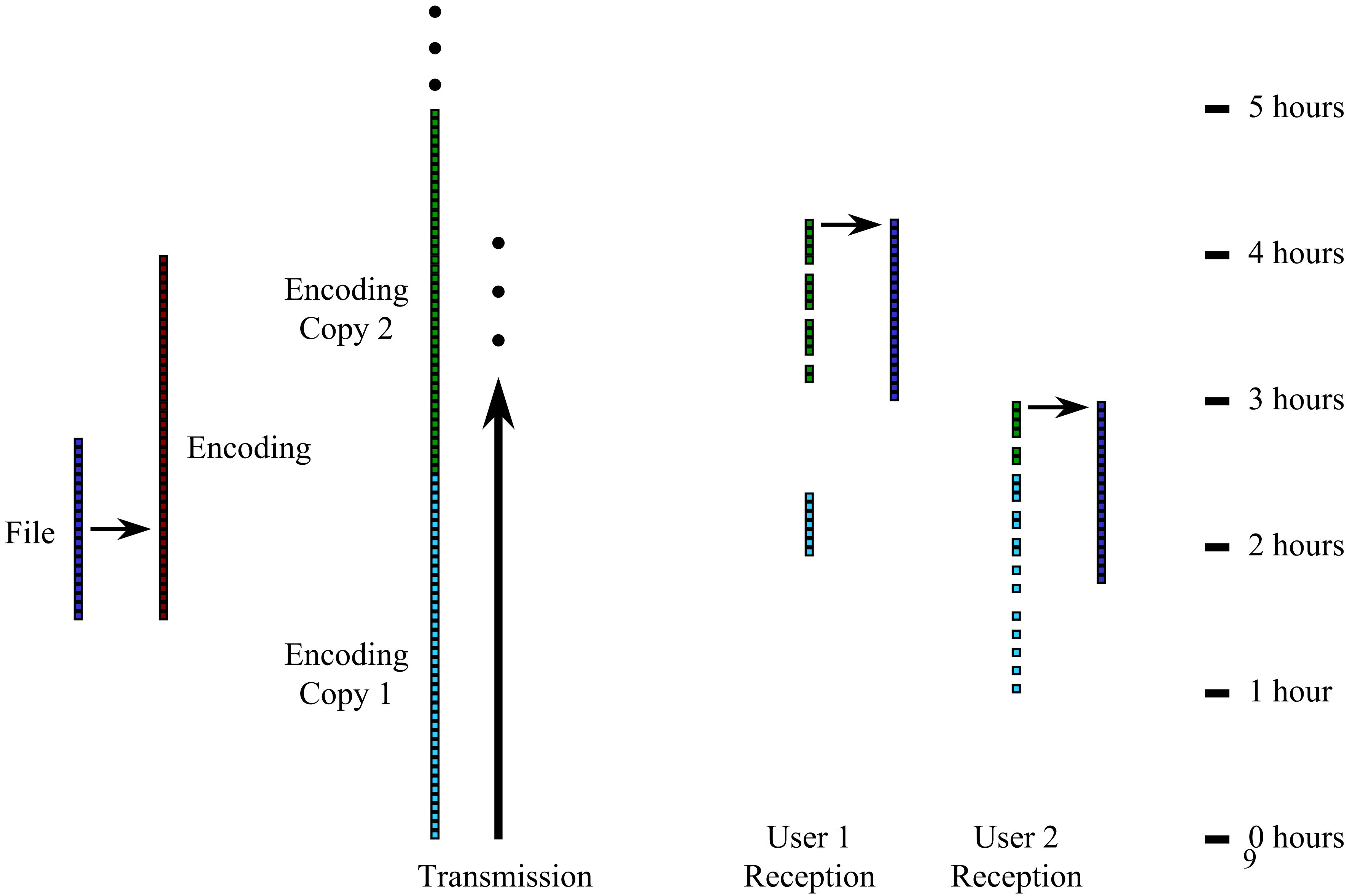
Broadcast Solution Features

- Bad
  - Users cannot initiate the download at their discretion.
  - Users cannot continue download seamlessly after temporary interruption.
  - Packet loss is a problem.
- Good
  - Low server load.
  - Low network load.
  - Does scale well.

A Coding Solution: Assumptions

- We can take a file of n packets, and encode it into cn encoded packets.
- From any set of n encoded packets, the original message can be decoded.

Coding Solution



Coding Solution Features

- Users can initiate the download at their discretion.
- Users can continue download seamlessly after temporary interruption.
- Moderate packet loss is not a problem.
- Low server load - simple protocol.
- Does scale well.
- Low network load.

So, Why Aren't We Using This...

- Encoding and decoding are slow for large files -- especially decoding.
- So we need fast codes to use a coding scheme.
- We may have to give something up for fast codes...

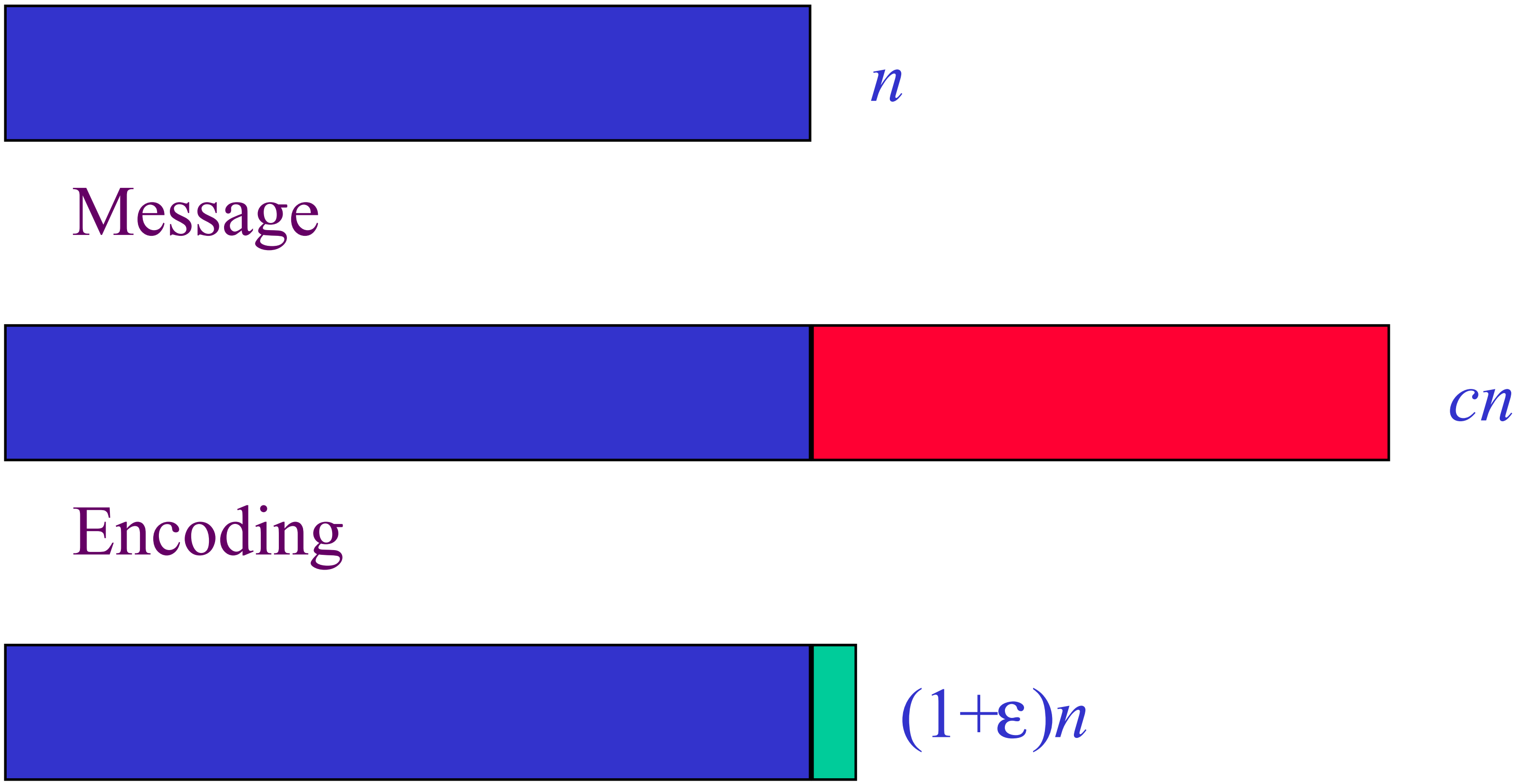
Performance Measures

- Time Overhead
  - The time to encode and decode expressed as a multiple of the encoding length.
- Reception efficiency
  - Ratio of packets in message to packets needed to decode. Optimal is 1.

Reception Efficiency

- Optimal
  - Can decode from any n words of encoding.
  - Reception efficiency is 1.
- Relaxation
  - Decode from any $(1+\varepsilon)\ n$ words of encoding
  - Reception efficiency is $1/(1+\varepsilon)$.

Parameters of the Code



Reception efficiency is $1/(1+\varepsilon)$

Previous Work

- Reception efficiency is 1.
  - Standard Reed-Solomon
    - Time overhead is number of redundant packets.
    - Uses finite field operations.
  - Fast Fourier-based
    - Time overhead is $\ln^2 n$ field operations.
- Reception efficiency is $1/(1+\varepsilon)$.
  - Random mixed-length linear equations
    - Time overhead is $\ln(1/\varepsilon)/\varepsilon$.

Tornado Code Performance

- Reception efficiency is $1/(1+\varepsilon)$.
- Time overhead is $\ln(1/\varepsilon)$.
- Simple, fast, and practical.

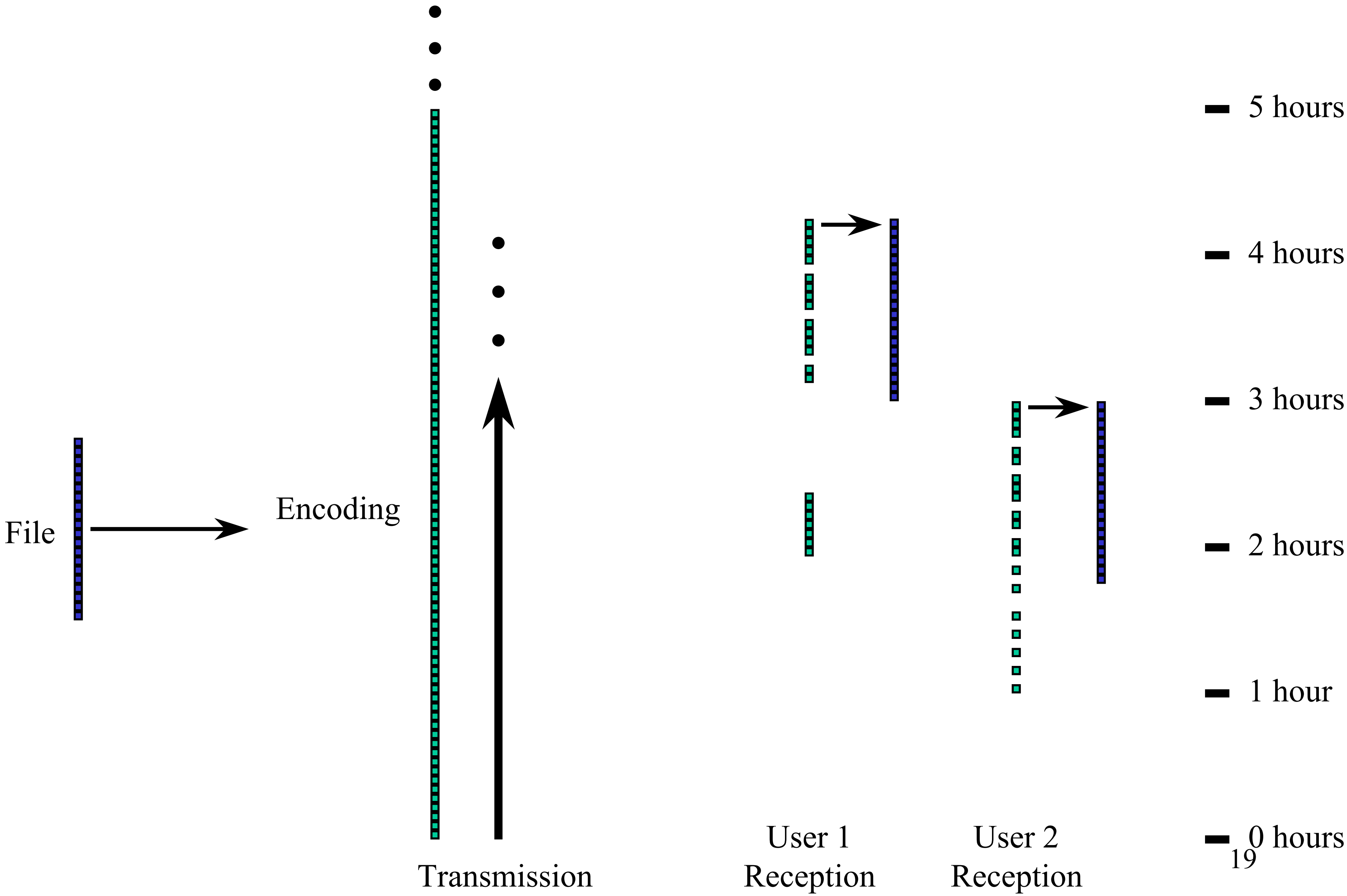
Codes: Other Applications?

- Using codes, data is like water.
- What more can you do with this idea?
- Example --Parallel downloads:
  Get data from multiple sources, *without the need for co-ordination*.

Latest Improvements

- Practical problem with Tornado code: encoding length
  - Must decide a priori -- what is right?
  - Encoding/decoding time/memory proportional to encoded length.
- Luby transform:
  - Encoding produced "on-the-fly" -- no encoding length.
  - Encoding/decoding time/memory proportional to message length.

Coding Solution



Bloom Filters: High Level Idea

- Everyone thinks they need to know exactly what everyone else has. *Give me a list of what you have.*
- Lists are long and unwieldy.
- Using Bloom filters, you can get small, approximate lists. *Give me information so I can figure out what you have.*

Lookup Problem

- Given a set $S = \{x_1, x_2, x_3, \ldots x_n\}$ on a universe U, want to answer queries of the form:

$$Is\ y \in S.$$

- Example: a set of URLs from the universe of all possible URL strings.
- Bloom filter provides an answer in
  - "Constant" time (time to hash).
  - Small amount of space.
  - But with some probability of being wrong.

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.

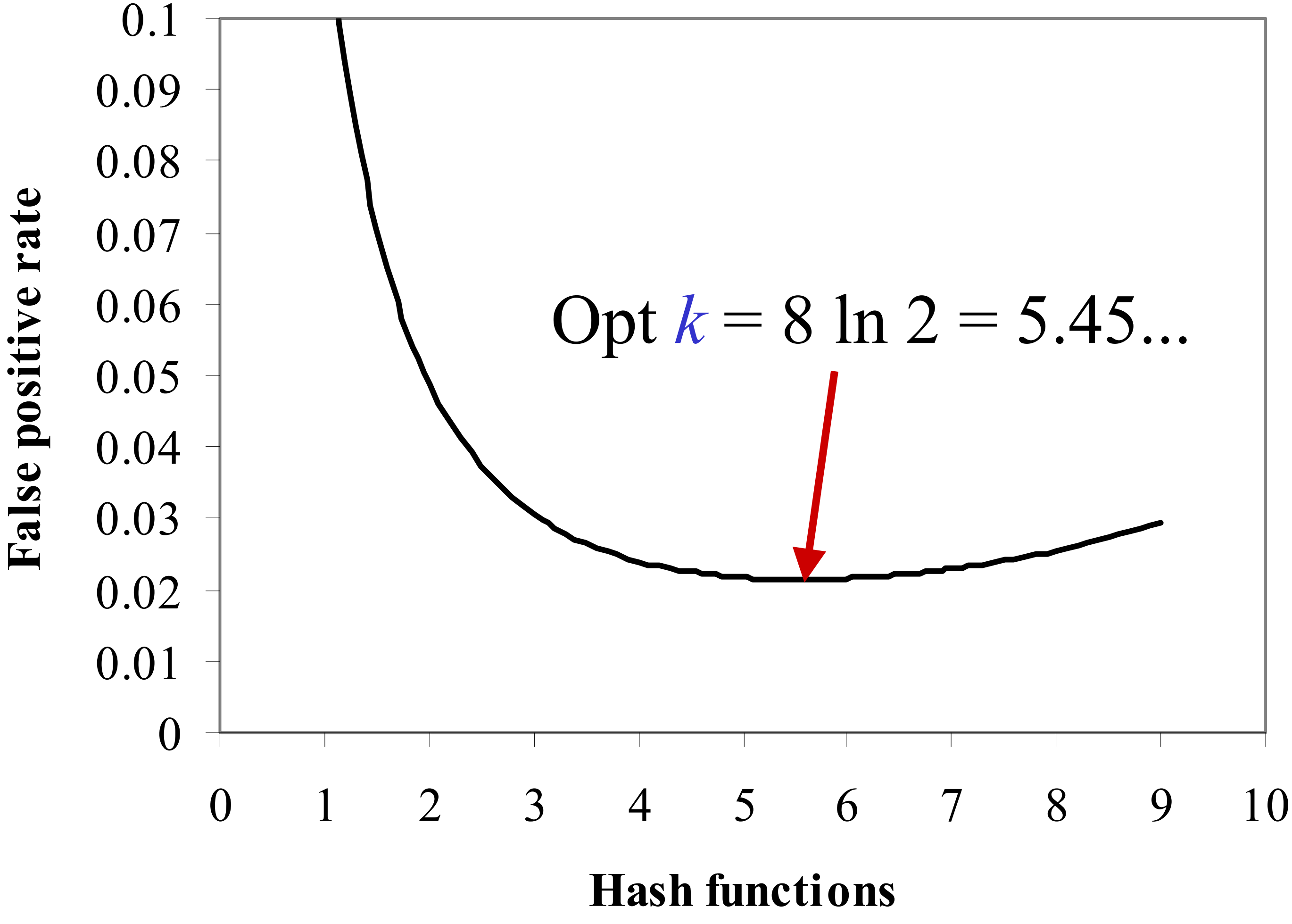
B

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

Errors

- Assumption: We have good hash functions, look random.
- Given m bits for filter and n elements, choose number k of hash functions to minimize false positives:
  - Let $p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$
  - Let $f = \Pr[\text{false pos}] = (1 - p)^k \approx (1 - e^{-kn/m})^k$
- As k increases, more chances to find a 0, but more 1's in the array.
- Find optimal at $k = (\ln 2)m/n$ by calculus.

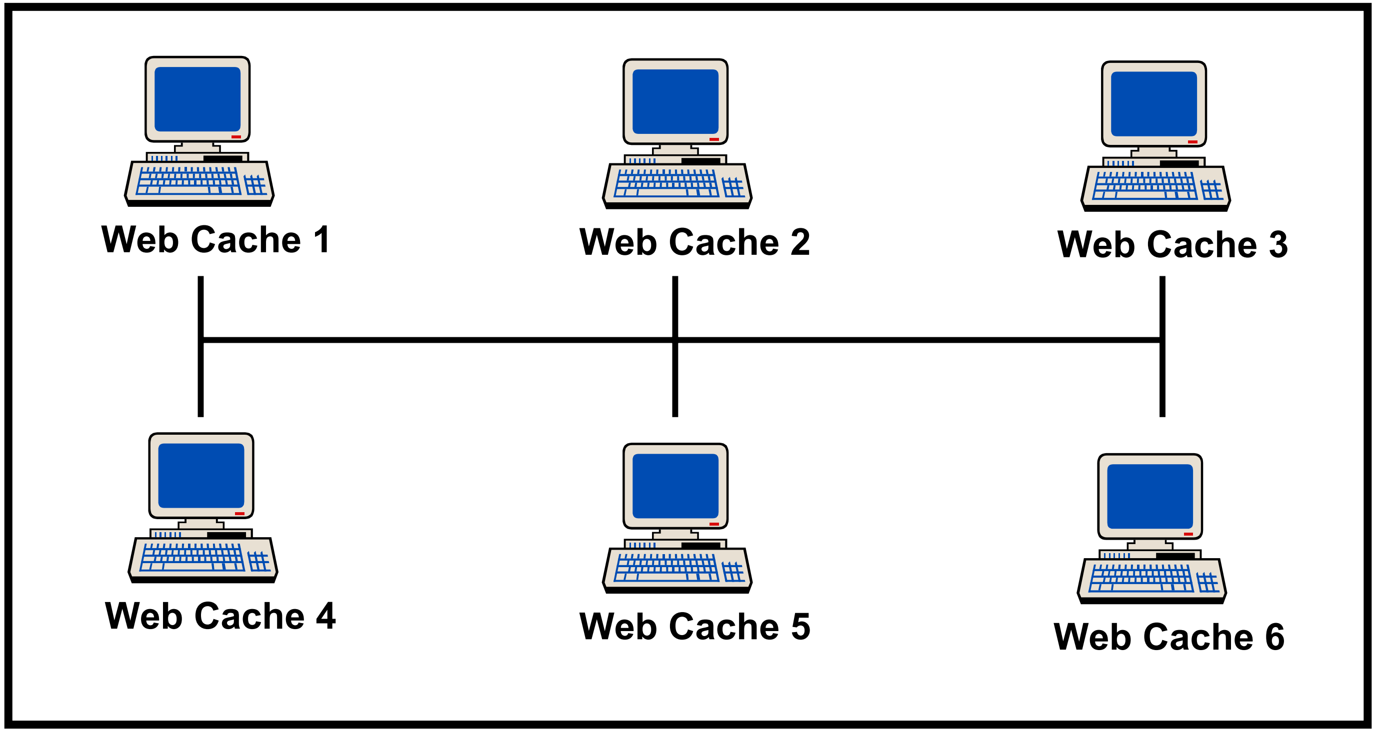
Example



False positive rate
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Opt $k = 8 \ln 2 = 5.45...$
0
1
2
3
4
5
6
7
8
9
10
Hash functions

$m/n = 8$

Bloom Filters: Distributed Systems



- Send Bloom filters of URLs.
- False positives do not hurt much.
  - Get errors from cache changes anyway.

Tradeoffs

- Three parameters.
  - Size m/n : bits per item.
  - Time k : number of hash functions.
  - Error f : false positive probability.

Compression

- Insight: Bloom filter is not just a data structure, it is also a message.
- If the Bloom filter is a message, worthwhile to compress it.
- Compressing bit vectors is easy.
  - Arithmetic coding gets close to entropy.
- Can Bloom filters be compressed?

Optimization, then Compression

- Optimize to minimize false positive.

$$p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$$

$$f = \Pr[\text{false pos}] = (1 - p)^k \approx (1 - e^{-kn/m})^k$$

$$k = (m \ln 2)/n \text{ is optimal}$$

- At $k = m\ (\ln 2)\ /n$, $p = 1/2$.
- Bloom filter looks like a random string.
  - Can't compress it.

Tradeoffs

- With compression, four parameters.
  - Compressed (transmission) size z/n : bits per item.
  - Decompressed (stored) size m/n : bits per item.
  - Time k : number of hash functions.
  - Error f : false positive probability.

Does Compression Help?

- Claim: transmission cost limiting factor.
  - Updates happen frequently.
  - Machine memory is cheap.
- Can we reduce false positive rate by
  - Increasing decompressed size (storage).
  - Keeping transmission cost constant.

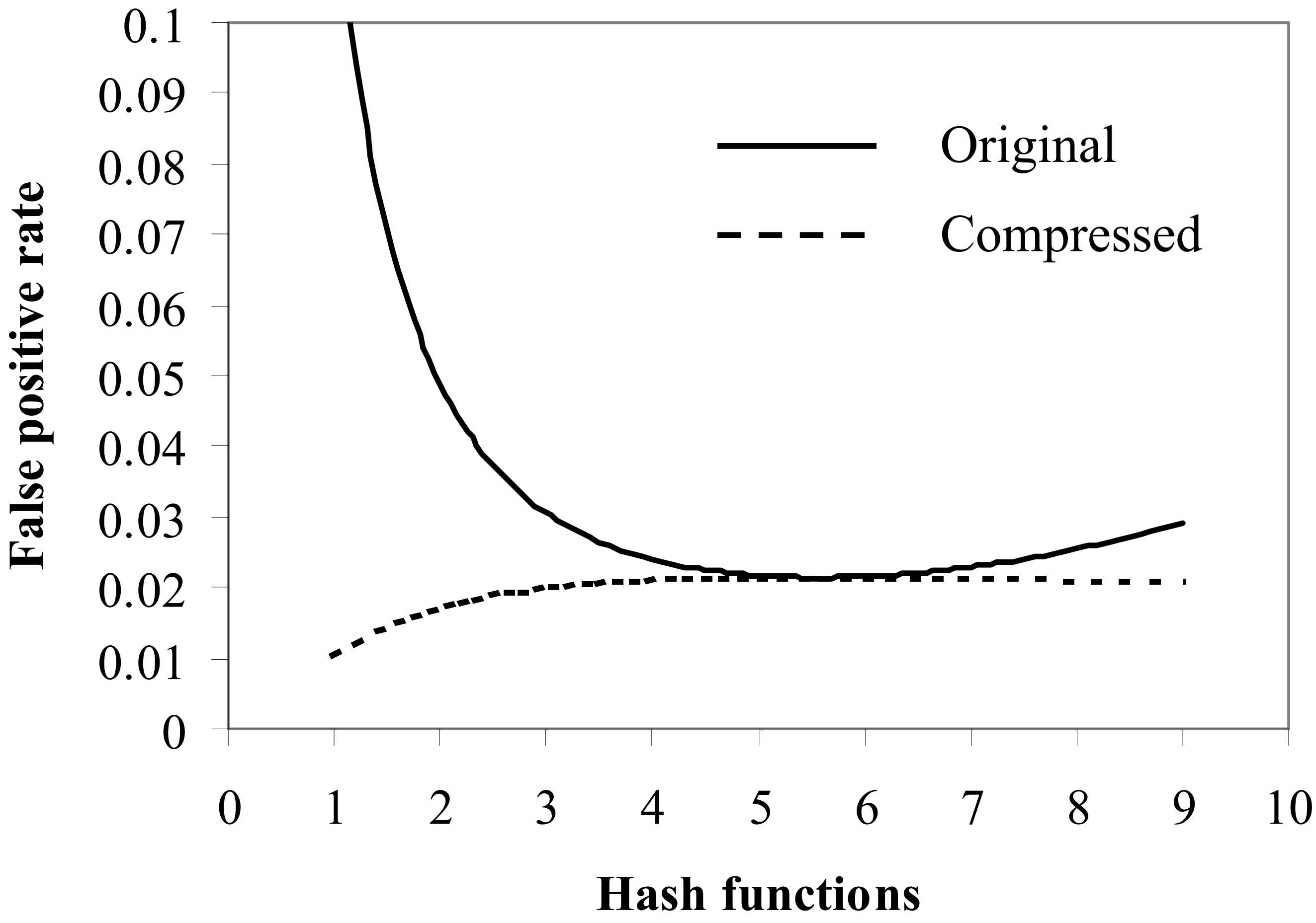
Errors: Compressed Filter

- Assumption: optimal compressor, $z = m\mathrm{H}(p)$.
  - $\mathrm{H}(p)$ is entropy function; optimally get $\mathrm{H}(p)$ compressed bits per original table bit.
  - Arithmetic coding close to optimal.
- Optimization: Given z bits for compressed filter and n elements, choose table size m and number of hash functions k to minimize f.

$$p \approx e^{-kn/m}; f \approx (1-e^{-kn/m})^k; z \approx mH(p)$$

- Optimal found by calculus.

Example



0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0 1 2 3 4 5 6 7 8 9 10
False positive rate
Hash functions
Original
Compressed

$z/n = 8$

Results

- At $k = m (\ln 2) / n$, false positives are maximized with a compressed Bloom filter.
  - Best case without compression is worst case with compression; compression always helps.
- Side benefit: Use fewer hash functions with compression; possible speedup.

Examples

Array bits per elt.	m/n	8	14	92
Trans. Bits per elt.	z/n	8	7.923	7.923
Hash functions	k	6	2	1
False positive rate	f	0.0216	0.0177	0.0108
Array bits per elt.	m/n	16	28	48
Trans. Bits per elt.	z/n	16	15.846	15.829
Hash functions	k	11	4	3
False positive rate	f	4.59E-04	3.14E-04	2.22E-04

- Examples for bounded transmission size.
  - 20-50% of false positive rate.
- Simulations very close.
  - Small overhead, variation in compression.

Examples

Array bits per elt.	m/n	8	12.6	46
Trans. Bits per elt.	z/n	8	7.582	6.891
Hash functions	k	6	2	1
False positive rate	f	0.0216	0.0216	0.0215
Array bits per elt.	m/n	16	37.5	93
Trans. Bits per elt.	z/n	16	14.666	13.815
Hash functions	k	11	3	2
False positive rate	f	4.59E-04	4.54E-04	4.53E-04

- Examples with fixed false probability rate.
  - 5-15% compression for transmission size.
- Matches simulations.

Bloom Filters: Other Applications?

- Finding objects
  - Oceanstore : Object Location
  - Geographical Region Summary Service
- Data summaries
  - IP Traceback
- Reconciliation methods
  - Coming up...

Putting it all Together: Informed Content Delivery on Overlay Networks

- To appear in SIGCOMM 2002.
- Joint work with John Byers, Jeff Considine, Stan Rost.

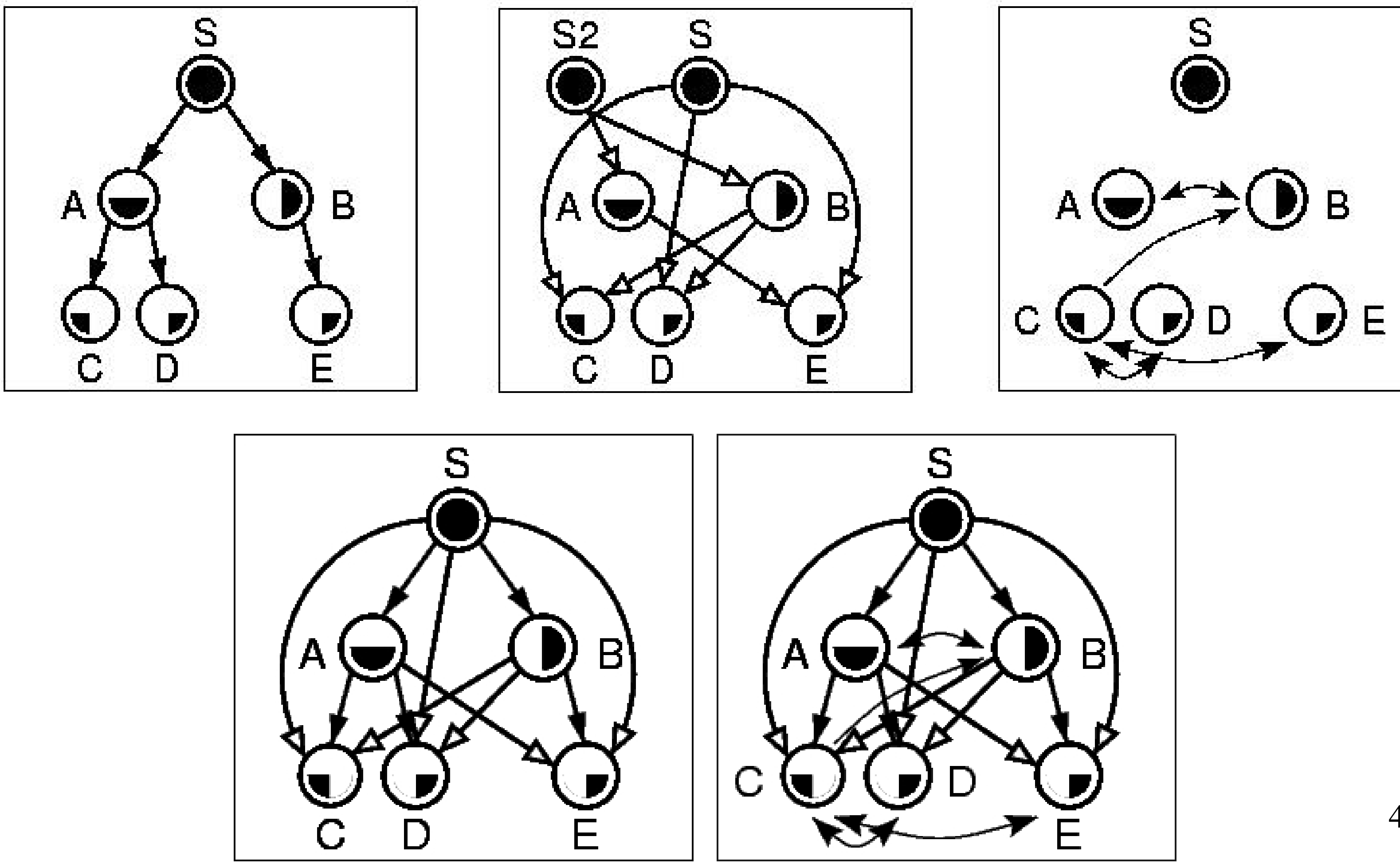
Informed Delivery: Basic Idea

- Reliable multicast uses tree networks.
- On an overlay/P2P network, there may be other bandwidth/communication paths available.
- But I need coordination to use it wisely.

Application: Movie Distribution Problem

- Millions of users want to download a new movie.
  - Or a CDN wants to populate thousands of servers with a new movie for those users.
- Big file -- for people with lots of bandwidth.
- People will being using P2P networks.

Motivating Example



Our Argument

- In CDNs/P2Ps with ample bandwidth, performance will benefit from additional connections
  - If intelligent in collaborating on how to utilize the bandwidth
- Assuming a pair of end-systems has not received *exactly the same* content, it should *reconcile* the differences in received content

It's a Mad, Mad, Mad World

- Challenges
  - Native Internet
    - Asynchrony of connections, disconnections
    - Heterogeneity of speed, loss rates
    - Enormous client population
    - Preemptable sessions
    - Transience of hosts, routers and links
  - Adaptive overlays
    - In reconfiguring topologies, exacerbate some of the above

Environmental Fluidity Requires Flexible Content Paradigms

- Expect frequent reconfiguration
  - Need scalable migration, preemption support
- Digital fountain to the rescue
  - Stateless: servers can produce encoded continuously
  - Time-invariant in memoryless encoding
  - Tolerance to client differences
  - Additivity of fountains

Environmental Fluidity Produces Opportunities

- Opportunities for reconciliation
  - Significant discrepancies between working sets of peers receiving identical content
    - Receiver with higher transfer rate or having arrived earlier will have more content
    - Receivers with uncorrelated losses will have gaps in different portions of their working sets
  - Parallel downloads
  - Ephemeral connections of adaptive overlay networks

Reconciliation Problem

- With standard sequential ordering, reconciliation is not (necessarily) a problem.
- Using coding, must reconcile over a potentially large, unordered universe of symbols (using Luby's improved codes).
  - How to reconcile peers with partial content in an informed manner?

Approximate Reconciliation with Bloom Filters

- Send a (compressed) Bloom filter of encoding packets held.
- Respondent can start sending encoding packets you do not have.
- False positives not so important.
  - Coding already gives redundancy.
  - You want useful packets as quickly as possible.
- Bloom filters require a small number of packet.

Additional Work

- Coarse estimation of overlap in 1 packet.
  - Using sampling.
  - Using min-wise independent samples.
- Approximate reconciliation trees.
  - Enhanced data structure for when the number of discrepancies is small.
  - Also based on Bloom filters.
- Re-coding.
  - Combining coded symbols.

Reconciliation: Other Applications

- Approximate vs. Exact Reconciliation
  – Communication complexity.
- Practical uses:
  – Databases, handhelds, etc.

Public Relations: Latest Research (1)

- A Dynamic Model for File Sizes and Double Pareto Distributions
  - A generative model that explains the empirically observed shape of file sizes in file systems.
  - Lognormal body, Pareto tail.
  - Combines multiplicative models from probability theory with random graph models similar to recent work on Web graphs.

Public Relations: Latest Research (2)

- Load Balancing with Memory
  - Throw n balls into n bins.
  - Randomly: maximum load is $\log n / \log \log n$
  - Best of 2 choices: $\log \log n / \log 2$
- Suppose you get to "remember" the best possibility from the last throw.
  - 1 Random choice, 1 memory: $\log \log n / 2 \log \tau$
  - Queueing variations also analyzed.

Public Relations: Latest Research (3)

- Verification Codes
  - Low-Density Parity-Check codes for large alphabets: e.g. 32-bit integers, and random errors.
  - Simple, efficient codes.
    - Linear time.
    - Based on XORs.
  - Performance: better than worst-case Reed-Solomon codes.
  - Extended to additional error models (code scrambling).

Conclusions

- I'm interested in network problems.
- There are lots of interesting problems out there.
  - New techniques, algorithms, data structures
  - New analyses
  - Finding the right way to apply known ideas
- I'd love to work with MIT students, too.